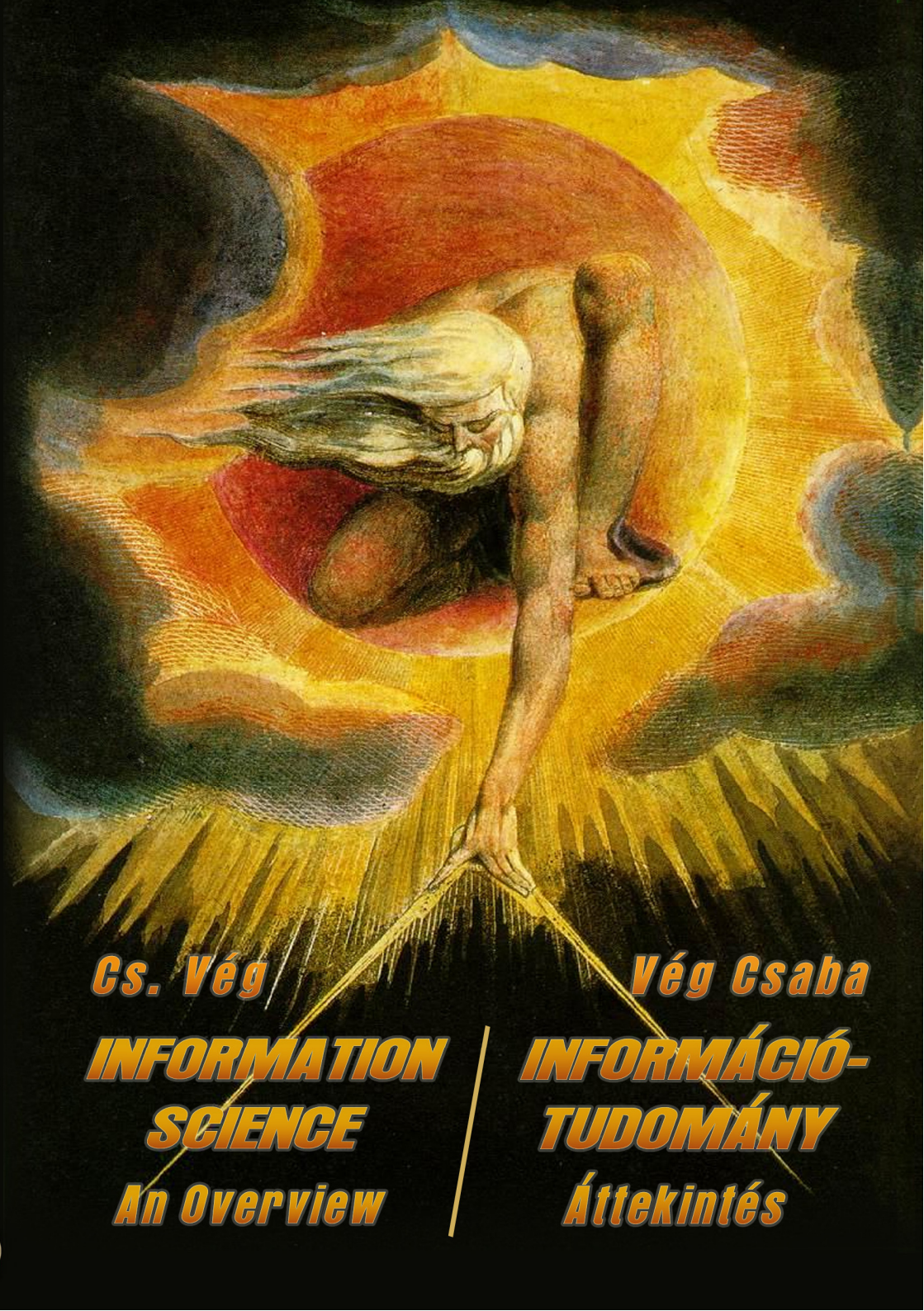


Cs. Vég

INFORMATION SCIENCE

An Overview

Vég Csaba

INFORMÁCIÓ-TUDOMÁNY

Áttekintés

Informatics and especially the development of information systems may be thought of as a catalyzer that, through its problems and toolsets, helps in the construction of a sound ideology for the representation and acquisition of information. The goal is to briefly define a theoretical approach, which once could be the basis of constructing an exact conceptual framework for software development, as well as of establishing the most fundamental concepts of generic information acquisition and representation, all with mathematical accuracy.

Az informatikát, azon belül is az információs rendszerek fejlesztését tekinthetjük egy olyan katalizátornak, amely problémáival, illetve eszközkészletével segít kialakítani egy tiszta fogalomrendszert az információ ábrázolásáról, illetve az ismeretszerzésrõl. A cél egy olyan elvi megközelítés felvázolása, amely egyszer elvezethet a rendszerfejlesztés egzakt fogalmi kereteinek kialakításához, valamint az általános értelemben vett ismeretszerzés és reprezentáció legalapvetõbb fogalmainak matematikai precizitású tisztázásához.

CSABA VÉG

INFORMATION SCIENCE
An Overview

VÉG CSABA

INFORMÁCIÓTUDOMÁNY
Áttekintés

Budapest, 2011.

English translation by / Fordította

Gusztáv Jánvári, Imprestige Ltd.

Cover Illustration / A borítón

William Blake: The Ancient of Days (1794)

Informatics and especially the development of information systems may be thought of as a catalyzer that, through its problems and toolsets, helps in the construction of a sound ideology for the representation and acquisition of information. The goal is to briefly define a theoretical approach, which once could be the basis of constructing an exact conceptual framework for software development, as well as of establishing the most fundamental concepts of generic information acquisition and representation, all with mathematical accuracy.

Az informatikát, azon belül is az információs rendszerek fejlesztését tekinthetjük egy olyan katalizátornak, amely problémáival, illetve eszközkészletével segít kialakítani egy tiszta fogalomrendszert az információ ábrázolásáról, illetve az ismeretszerzésről. A cél egy olyan elvi megközelítés felvázolása, amely egyszer elvezethet a rendszerfejlesztés egzakt fogalmi kereteinek kialakításához, valamint az általános értelemben vett ismeretszerzés és reprezentáció legalapvetőbb fogalmainak matematikai precizitású tisztázásához.

Notations

- Definitions are **bold**.
- Statements and short examples are enclosed in brackets. Relevant statements are written in *italic* or ***bold-italic*** font.

Mathematical notations:

- Relations of the nature of "if and only if" are expressed using the *iff* phrase.
- The set of natural numbers consists of 0 and its integers increments.
- $a_{i \in I}$ is the simplified presentation of the indexed elements a_i and an index set I, where "i iterates over the values of I". In case of operations, e.g., for $\bigcup_{i \in I}$, traditional interpretation is assumed. The example, therefore, should be interpreted as "i iterates over the elements of index set I one by one".

Jelölések

- A definíciókat **vastagított** („félkövér”) betű jelöli.
- Az állítások és rövid példák zárójelben szerepelnek. A fontosabb állítások *dőltbetűvel* vagy ***dőlt-vastagított*** betűvel vannak jelölve.

Matematikai jelölések:

- Az „akkor és csak akkor, ha” állításokat a *„pontosan akkor, ha”* szókapcsolat írja le.
- Természetes számnak a 0 számot és rákövetkezőit tekintjük.
- Az $a_{i \in I}$ az egyszerűsített jelölése az indexelt a_i elemeknek, valamint egy I indexhalmaznak, ahol az i sorra felveszi az I értékeit. Műveletek, pl. $\bigcup_{i \in I}$ esetén a hagyományos értelmezést használjuk, tehát ebben a példában: „i sorra felveszi I indexhalmaz elemeit”.

Contents

Tartalom

CSABA VÉG

INFORMATION SCIENCE
An Overview

Informatics and especially the development of information systems may be thought of as a catalyzer that, through its problems and toolsets, helps in the construction of a sound ideology for the representation and acquisition of information. The goal is to briefly define a theoretical approach, which once could be the basis of constructing an exact conceptual framework for software development, as well as of establishing the most fundamental concepts of generic information acquisition and representation, all with mathematical accuracy.

Information Science

Theoretical Approach

The book titled "Feljegyzések…"[1] is an initiative to provide definitions of the most fundamental concepts of Information Science. The followings provide an overview of the most important elements of this initiative. The exact mathematical definitions are included in the referenced book.

What practical advantages may the development of a theoretical approach yield in when it comes to software development and Information Technology? Information Technology here is defined as the act of *capturing, representing and transforming information regarding various systems*, including representations in the forms of computer programs or applications. What results can the development of such a conceptual system produce?

Taking a conceptual perspective, with the help of an appropriate conceptual approach, the study field of Information Science and, partially, that of Information Technology can be defined more accurately, namely as the set of fundamental topics on capturing, representing and transforming information. For example, the development of an information system first includes gathering requirements (that is, capturing information or acquiring knowledge), then representing those as draft models (e.g., analysis, design), as well as computer programs, but it also covers the transformations and modifications of the mentioned representations. The running system can also be interpreted as knowledge acquisition (through a user interface, e.g.) of external events and data, and as representation and transformation of knowledge. With an exact conceptual background, it can be precisely defined what IT specialists do during their work or what a certain behavior sequence of a system means.

Since our conceptual system discusses the definition, representation and transformation of knowledge in a general sense, the results can also be

[1] Vég Csaba: Feljegyzések az információtudomány elméleti alapjairól (2008. http://mek.oszk.hu/06300/06303)

applied for areas outside of Information Technology (measurements, experiments, everyday events, etc.).

From a practical and technological aspect the theoretical foundation regarding the representation of information would allow for defining high-level notation systems. Current programming languages and modeling notation systems (such as OMT, UML, ER diagrams and so on) told "high-level" are all developed from the machine logic of computers following a bottom-up approach. Developments were driven by the goal to simplify the series of instructions of some given nature. Thus, this progress has proven to be occasional, as certain techniques have been realized, while others have not. A solid theoretical foundation, however, enables for notation languages to be developed top-down and for determining the required and satisfying or minimal (!) set of notations. This could be the basis of "information languages" featuring a level of expressiveness close to those of human (technical) languages. This can be roughly imagined as if an appropriate set of document-like artifacts such as requirement specifications and design documents would result in a ready-to-run, implemented software system. Numerically expressed, these represent an abstraction level of the order of 1-2 magnitudes higher, thus in case of such languages, a single line corresponds to 20-200 lines of "high-level" program code.

A theoretical approach would also enable the development of special "information systems", which users can tell data, events or even rules relating to the behavior of the system in a way similar to that of a human language (even expressing things orally), and the system would automatically modify its internal states or rules, by locating the locations of definitions of one or the other piece of the related information. Simpler versions of such systems are the reaction/collaboration-based descriptions and systems, the parts of which are capable to track events of the external world, reacting to those, as well as collaborating with further systems in an "intelligent" (but, of course, programmed) way. This method also simplifies the definition of business applications and can also be applied, for example, to robot control.

An appropriate theoretical conceptual system is also suitable for defining the scope of usability of various general-purpose frameworks, generators or even notations.

I.

System, Abstraction and Composition

System

Fundamental goals of Information Science are defined in this book as the acquisition, representation and transformation of information. Information, however, is never without *regarding* something that this book calls *system*. The only observable properties of a system, according to our principle, are its *state* and the *change* of its state, therefore systems appear as *processes*.

The approach described in this book employs a kind of innovation in that systems (defined by general specifications) and their *observed* changes are handled the same way. Observation appears as *uncertainty*, that is, in each point of time (called moments), only one set of states can be determined (for example, one can determine that temperature is between 36 and 37 Celsius degree, but does not know any more accurate value). Another innovation our approach features is that the system is considered in its *completeness*: all states regard the entire system, enabling states to be complex, however.

That is, for a **system (of general specification)** $\sigma = (\mathcal{T}, S, P)$, we define the moments (with a non-empty ordered set T), the possible states (with a non-empty set *S*) and the function $P : \mathcal{T} \to 2^S \setminus \{\varnothing\}$ of the process assigning a (non-empty) set of states to each moment. The set of states is called a state combination and is interpreted as the states the system rest in at certain moments. This definition states nothing special but that a system may be in a state of several possible states at any given moments. Since we do not exactly know which state the system rests in, we can only specify a set of states or a combination of states. Upon knowing nothing about the state at a given moment, the state combination consists of the set of all possible states

($P(t)=S$) and it is said the system is in an **uncertain, arbitrary state**. Moments in which the system is not in arbitrary state are collectively called **time of the system** (noted by T, where $T \subseteq \mathcal{T}$). The original set of moments $\mathcal{T}$ actually denotes the time of the observer.

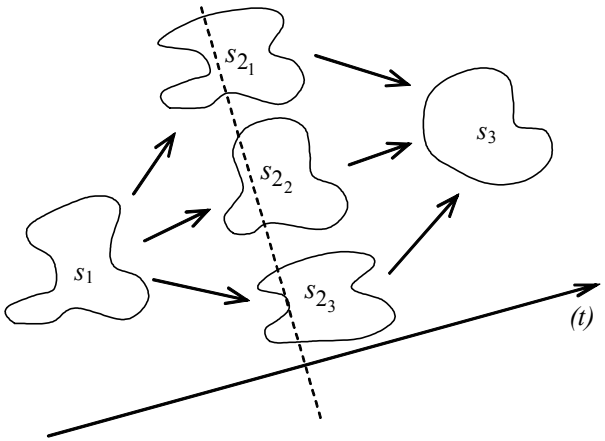


Figure 1: Systems defined by general specification may be in one of several states in every moment.

Certain investigations become simpler when the system is defined in form of $\sigma = (\mathcal{T}, T, S, P)$, called **deterministic specification**, which, apart from the time and the possible states, also accounts for the time of the system and, in addition, defines the change function in the form of $P : T \to S$, which assigns each moment $t \in T \subseteq \mathcal{T}$ exactly one state.

Should some given systems $\sigma_{i \in I} = (\mathcal{T}, S, P_i)$ share an identical set of moments and an identical set of states, those systems are considered **direct comparable** (noted as $\Sigma^{(\sigma_i)}$, while systems direct comparable with a system σ are noted as $\Sigma^{(\sigma)}$), since the relations (such as the size of the state sets) between the state combinations the systems rest in can be investigated in every moment.

A system may be in uncertain state in every moment (or all the time). Such systems are called **arbitrary abstract systems** and are noted by $\aleph$.

It is also possible that a system σ and another system σ' have different sets of time and states, but show changes of the same *nature*, i.e., there is a strongly monotonic (that is, one producing changes in the same order) invertible function $\alpha_{\equiv_T} : T \to T'$ for time and an invertible function $\alpha_{\equiv_S} : \overline{S} \to \overline{S}'$ for states, which leave the subset-relations of state combinations of (possibly observed) changes intact, that is, which do not modify the *essence of change*. Such systems are considered **equivalent** ($\sigma \equiv \sigma'$) or, in other words, one system is the **renotation** of the other. (It is the renotation of the system of temperature change measured in Celsius degrees when values are specified in Fahrenheit or Kelvin degrees. These systems are equivalent.)

(All arbitrary abstract systems are equivalent.)

The function pair $\alpha_{\equiv} = (\alpha_{\equiv_T}, \alpha_{\equiv_S})$ used to convert moments and state combinations is called **direct renotation**. A direct renotation relates two groups of systems with identical sets of time and states (therefore, direct comparable), that is, direct renotation specifies the equivalent renotations of a given system, which, therefore, show changes of the same nature.

System states may be complex consisting of a generic Cartesian product of more atomic state sets. This allows for defining the **composition of systems** defined in the same time $\mathcal{T}$ such that initial systems are considered parts, and it is specified which part regards which location. **Parts** are called **components** or **subsystems**.

A system (that is, the described process) does not change *essentially* if the set of time $\mathcal{T}$ is extended with moments in which the function undefined, or such moments are subtracted from the set. Similarly, a composite system remains *essentially* intact when components with undefined states in given moments are added or subtracted. Such "essentially identical" systems are called **equatable systems** (noted as $\sigma \Leftrightarrow \sigma'$; while systems equatable to a system σ are noted by $\sigma^{\Leftrightarrow}$).

(Equatable systems are equivalent. All arbitrary abstract systems are equatable.)

Comparable Systems

By adding moments to some given systems in which the state of the systems is undefined, or removing such moments, the systems may change to have the same set of moments (time) and states. Such systems become (direct) comparable, therefore they are called (indirectly) **comparable systems** (noted as $\Sigma^{(\sigma_i)^*}$; while the set of systems comparable to a system σ is noted as $\Sigma^{(\sigma)^*}$).
(A composition and its components are comparable.)

State combinations taken at a given moment and therefore the entire processes of state changes of comparable systems are also comparable, and several other operations may be interpreted on such systems, too.

How can systems be compared? If, for example, two systems do not share any elements of their state combinations at a given moment (that is, at the time they may rest only in different states), those systems are treated **contradictory**.

(No system is contradictory to the arbitrary abstract system.)

A system σ_A, moreover, may specify an identical or extended state combination at *every* moment than another noted by σ_B. In this case σ_A is the **consequence** of σ_B (noted as $\sigma_A \Leftarrow \sigma_B$) and, reversely, σ_B is the **condition** or **example** of σ_A (noted as $\sigma_B \Rightarrow \sigma_A$).

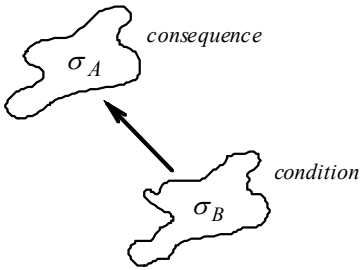


Figure 2: Consequence and condition.

(The arbitrary abstract system is the consequence of every system, and all systems are conditions of the arbitrary abstract system. Two systems are

conditions and consequences of each other ($\sigma_A \Leftarrow \sigma_B$ and $\sigma_A \Rightarrow \sigma_B$) if and only if they are equatable ($\sigma_A \Leftrightarrow \sigma_B$). Each system is the condition and, jointly, the consequence of itself. Every system is the consequence of its condition (example).

If a consequence of a system is other than the arbitrary abstract system, that consequence is called the system's **restrictive constrainer**, or **constrainer** for short.

Consequences of a system, that is, when the combinations of the possible states for certain moments are extended (including the case of ignoring some moments, this way increasing the uncertainty in the system) are the **comparable abstractions** of the initial system. The condition of a system is, in other words, the **comparable specialization** of the mentioned system. A comparable specialization (i.e., condition or example) of a system is an **instance** of the system if that specialized system takes exactly one state in each of the moments the initial system is defined in.

Comparable systems may be the subjects of certain operations, namely set operations performed on state combinations taken in the various moments.

Conjunction may be calculated for uncontradictory comparable systems, based on the intersection of state combinations taken at the same moments. (The result of the conjunction is the condition of every initial system participating in the operation.)

Similarly, using the union operation, the **disjunction** of two or more (contradictory or uncontradictory) systems may be calculated, and the resulted system will be the consequence of every initial system participating in the operation.

Set operations also allow us to define the **converse** or **negation** of a system. A converse system is in state $P'(t) = S \setminus P(t)$ (i.e., "non-$P(t)$") in every moment (T) of the initial system.

(Arbitrary abstract systems are converses of themselves. Every system has a direct comparable converse. The union of a system and its direct comparable converse is the comparable arbitrary abstract system.)

Summarizing the comparison of comparable systems, a system may be a consequence or a condition of another, in which case these systems are **dependents** ($\sigma_A \wr \sigma_B$), or may be contradictory to each other, while, in the third case, the two systems are **independents**.

Abstraction, Specialization, Type

Equatable systems describe *essentially* identical change, while equivalent systems describe the relation of changes of the same *nature*. Condition yields in a more detailed description of the essence of changes, while consequence results in a more generic description of the essence of changes. Similarly, we can define the relation between more and less diverse changes, called specialization and abstraction, respectively. A system is the **abstraction of** some initial **systems** if it is the comparable abstraction (consequence) of the (equivalent) renotations of the initial systems. **Specialization of systems** can be defined similarly. Performing disjunction operations on (different) renotations of some initial systems is called **indeterministic abstraction**.

(Two systems are equivalent if and only if each one is both the generalization (abstraction) and the specialization of the other. Any system can be transformed to an arbitrary system by a specialization following an abstraction.)

Let's classify (i.e., let's group into disjoint subsets) the states (but not state combinations) of certain systems, then assign each class a state during a renotation and, in addition, disregard (ignore) certain moments of the initial systems. This yields a special abstraction called **type abstraction**. The result of the type abstraction is called **type**, **pattern** or **scheme**. (For example, learning temperature at only certain moments and by rounding values is a type abstraction.)

(The type abstraction assigns the consequence of the initial system the consequence of the results system.)

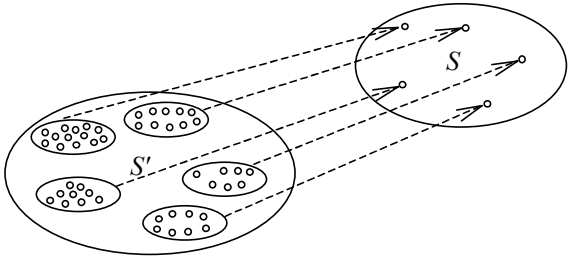


Figure 3: Type abstraction distributes initial states into classes.

Leaving a component off a composite (by replacing with a component resting in an arbitrary state in each moment) results in the consequence of the initial system. At this step we "abstracted" the ignored component or, in other words, "deleted" or "ignored" that component. This operation is called **simple abstraction**. The opposite transformation when a system is extended with one or more components is called **extension** or **simple specialization**.

(Any system equivalent to the result of any specialization may be defined by extension.)

II.

Information and Knowledge Acquisition

Information

All *constrainers* (i.e., consequences other than arbitrary abstract systems) are approximations of the system in the sense that each of those specifies the possible states the system may rest in at various moments (that is, its state combinations) at a certain granularity level or accuracy. Therefore, constrainers of a system are called (pieces of) **information regarding the system**.

This definition is not against traditional interpretation of information, where information is described as something new, something unexpected. A piece of information is called **novelty** or **(subjective) information** from the aspect of a consequence, if it is not the consequence of the mentioned consequence. (Based on the above, information is subjective information from the aspect of the arbitrary abstract system or, in other words, novelty in an "absolute aspect". Any piece of information regarding the system is the consequence of that system, that is, the initial system is a condition of that information.)

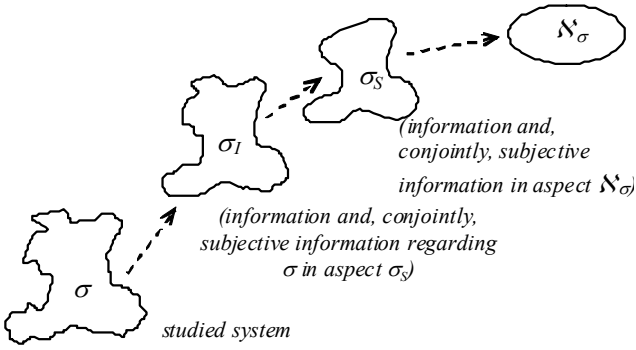


Figure 4: Information and subjective information

Representation

Up to this point, we have managed to define what is called information and what information regards (called system). But how can one define the *representation* of systems (and, accordingly, information) which connects represented *systems* with *representations*?

Based on earlier definitions, the function $\alpha_{\equiv} : \Sigma^{(\sigma)} \to \Sigma^{(\sigma')}$ operating between two sets of direct (!) comparable systems and mapping an initial system to a system equivalent to the initial one is called *direct renotation*. Enabling both the initial and the result systems to be transformed to comparable systems, direct renotation can be extended to be (an indirect) **renotation** ($A : \Sigma^{(\sigma_{i\in I})^*} \to \Sigma^{(\sigma_{j\in J})^*}$), connecting comparable (and not only direct comparable) systems.

Even *renotation* is an unnecessarily "strong" operation, since renotation has to be defined also for moments and state combinations that may not even appear in the systems $\sigma_{i\in I}$ and $\sigma_{j\in J}$ we want to connect. Usually accurate renotation is not a goal, as renotation of a certain granularity is just enough.

The function assigning a (comparable) consequence to each of some comparable systems is called **abstraction method** ($\alpha : \Sigma^{(\sigma)^*} \to \Sigma^{(\sigma)^*}$, $\alpha(\sigma') \Leftarrow \sigma' \in \Sigma^{(\sigma)^*}$). **Specialization method** may be defined similarly.

(An interesting attribute of abstraction methods is that applying those on comparable systems extends the scope of comparability: $\Sigma^{(\sigma_{i\in I})^*} \subseteq \Sigma^{(\alpha(\sigma_{i\in I}))^*} \subseteq \Sigma^{(\aleph)^*}$)

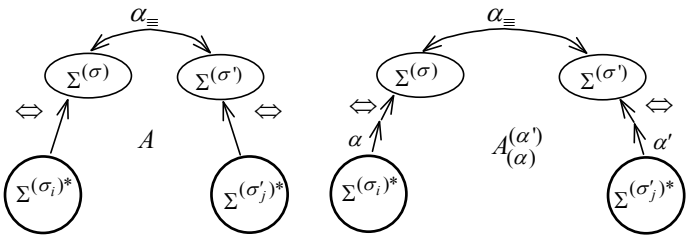


Figure 5: Renotation and abstract renotation

Based on the above, we can introduce the concept of **abstract renotation** as the construction consisting of the abstraction method α of

comparable systems $\Sigma^{(\sigma_{i\in I})^*}$, the abstraction method α' of comparable systems $\Sigma^{(\sigma'_{j\in J})^*}$ and the renotation $A:\Sigma^{(\alpha(\sigma_{i\in I}))^*}\to\Sigma^{(\alpha'(\sigma_{j\in J}))^*}$ (noted as $A_{(\alpha)}^{(\alpha')}:\Sigma^{(\sigma_i)^*}\to\Sigma^{(\sigma'_j)^*}$).

An abstract renotation $A_{(\alpha)}$ (provided that α' is trivial, i.e., it maps each system to one of its equivalent systems) may approximate a system to only a certain abstraction level. (Taking a digital speed display of a vehicle as an example, it maps speed ranges to numeric values and, therefore, reflects or approximates speed only to a certain level of abstraction or range.)

For an abstract renotation $A^{(\alpha')}$ (that may be, for example, the inverse of $A_{(\alpha)}$), images were able to approximate systems of greater granularity than the abstract renotations of the initial system, however images would have to refer to moments in which the system is not defined or to undefined state combinations. (Think of presenting a digital value using an analogue reader.)

Should abstraction methods α and α' both be trivial, A becomes an (indirect) renotation.

(An abstract renotation may be extended to be a renotation by extending $\alpha_{\equiv}$, while a renotation may be reduced to an abstract notation. Therefore any abstract renotation relies on renotations and renotations "contain" abstract renotations.)

Abstract renotations of the form $A_{(\alpha)}:\Sigma^{(\sigma_{i\in I})^*}\to\Sigma^{(\sigma_{j\in J})}$ generate outlined (brief) "renotations" o representations of the initial systems, and so are called **abstract representation methods**.

Once we choose an abstract representation method, any arbitrary result system will be called the **abstract representation of the** initial **system**. That is, anything can represent anything, but representation may only be studied when regarding a certain abstract renotation method and an initial system.

As per a remark above, an abstract representation method $A_{(\alpha)}$ may be extended to several renotations A enabling inversion, and therefore the image of an abstract representation, which image is comparable to the initial systems, may be determined. Let an arbitrary σ' be the abstract representation of an initial system σ. The image (i.e., the representation

yielded by an extension) $A(\sigma) \in \Sigma^{(\sigma_j)*}$ of an initial system is called (a) **base** of the abstract representation, while the result $A^{-1}(\sigma')$ applied on the abstract representation (what is the inverse image of the representation yielded by "back-renoting") is called the **approximation** of the initial system or **knowledge** about the system..

The representation of systems and information can be therefore defined by defining a method $\iota : \Sigma^{(\sigma_j)*} \to \Sigma^{(\sigma_j)}$ called **interpretation** for an abstract representation method $A_{(\alpha)} : \Sigma^{(\sigma_{i\in I})*} \to \Sigma^{(\sigma_{j\in J})}$. Interpretation assigns a system called **representation** or **denotation** an abstract representation, the **meaning** or **semantics** of that denotation.

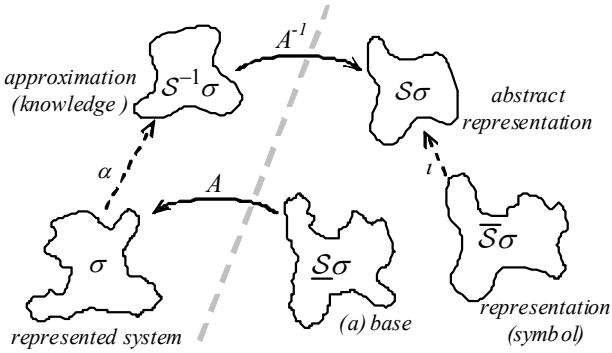


Figure 6: Abstract representation and representation.

Should an interpretation be invertible, a denotation maps to exactly one meaning and a meaning maps to exactly one denotation. Otherwise the preimage of a meaning is a set of denotations ($\iota^{-1}(\sigma') \subseteq \Sigma^{(\sigma_j)*}$), that is, the abstraction of possible denotations (as systems), called the **abstract denotation** of the meaning. Abstract denotation, hence, specifies a group of denotations of the same meaning. (The various visualizations of the digit 1, for example, constitute an abstract denotation.)

Let's choose a specialization method $\overline{\beta}$ for an abstract representation method $A_{(\alpha)}$ and an interpretation method ι, which selects exactly one

particular denotation from the abstract denotations (in other words, from all possible variations). Such a composite function $\overline{\beta}(\iota^{-1}(A_{(\alpha)}\sigma))$ is called **representation method**. Representation method determines the abstract denotation of the initial system to be represented, and selects a particular denotation, that is, an actual representation. A representation method specified by an abstract representation method $A_{(\alpha)}$ and an interpretation ι (with an arbitrary specialization method $\overline{\beta}$) is noted as $A_{(\alpha)}^{(\iota)}$, which therefore specifies representation methods up to the extent of the abstract denotation only.

An abstract renotation $A_{(\alpha)}^{(\alpha')}$ is a representation, too, called **simple representation**, where the interpretation is an abstraction method α', assigning representations their comparable abstractions as semantics (meanings).

Aspect, Accuracy and Difference

So, anything may represent anything. But how can we tell which representation is more accurate? For a given representation method, an **exact** or **accurate representation** of the system is the representation the semantics (meaning) of which is equatable to (any) bases of the representation (in which case bases are equatable).

(Renotation, i.e., the system image equivalent to the system, is an exact representation.)

Those representations the semantics of which are (non-trivial) consequences of any base are called **approximative representations**. Inexact and non-approximative representations are **erroneous representations**. **Exact**, **approximative** and **erroneous knowledge** (or approximation) can be defined similarly, by comparing the initial system with the given piece of knowledge.

Knowledge and representations may be qualified at a finer granularity if, apart from their relations, we can also define their differences. The difference of comparable systems, however, may vary in different *aspects*. Aspect is defined as a function assigning each initial system its more and more abstract consequences. The formal definition is as follows:

An **aspect** or **viewpoint** $\prec$ (pronounced as "v") of some comparable systems is a set of abstraction methods assigned to ordered values (with a given minimum $i0$ and a maximum $i\infty$) such that systems are mapped to their equatable systems at the mentioned minimum ($\prec_{(i0)}: \sigma_i \mapsto \sigma_i^{\Leftrightarrow}$), to the arbitrary abstract system at the maximum ($\prec_{(i\infty)}: \sigma_i \mapsto \aleph$) and, moreover, greater values are mapped to the consequences of the images of smaller values (for $a \leq b$, $\prec_{(a)}(\sigma) \Rightarrow \prec_{(b)}(\sigma)$, that is, abstraction increases).

(A series of abstraction methods performed a given number of times (including 0) in sequence is an aspect if the limit of the series is the abstraction producing the arbitrary abstract system.)

Aspect enables defining the **difference of systems** ($\Delta_{j\in J}^{(\prec)}\sigma_j$) by performing the abstraction methods on the systems and locating the smallest (index) value of the aspect for which its related abstraction yields equatable systems. The difference of systems is therefore an index value (a value) that can be compared with other differences in case of identical aspects. The **accuracy** of systems (e.g., a system and the knowledge regarding that system) is the system belonging to the difference of those systems (noted as $|\sigma_a, \sigma_b, ...|_{\prec}$). Accuracy is therefore the set of abstract images of the systems in a given aspect so that those abstract images are the first images identical to each other.

(Accuracy is the consequence of the initial systems. Accuracy is the arbitrary abstract system in case of maximal difference. The difference of equatable systems is minimal (*i0*), while their accuracy is equatable to those systems.)

(Comparable systems have accuracy in any aspect, and that accuracy is always an example of the arbitrary abstract system $\aleph$.)

Should we choose the set of extended non-negative real numbers (i.e., the set starting from zero and heading for infinite) or one of its appropriate subsets as the domain of the aspect, the ***difference in an aspect specifies a distance*** (!) in the scope of comparable systems, and therefore we **systems** may be said to be **more or less comparable to a given system in an aspect**. The requirements of the concept of distance are all satisfied: (1) The distance of two systems is non-negative, and is 0 if and only if the two systems are equatable. (2) The distance of σ_a and σ_b is equal to the distance of σ_b and

σ_a. (3) The sum of the distance of σ_a and σ_b and that of σ_b and σ_c is not greater than the distance of σ_a and σ_c.

(The consequence of a representation's semantics is the accuracy of the representation. An exact representation is more accurate than any inexact representation for any aspects. The accuracy of knowledge—as knowledge—is non-erroneous knowledge in any aspect.

(If a piece of knowledge amongst some dependent non-erroneous pieces of knowledge is more accurate in an aspect, than it is accurate at least to the extent the other pieces of knowledge are accurate in any (!) aspect.)

(Pieces of non-erroneous knowledge may not be contradictory systems.)

(***Considering a set of independent*** (i.e., non-erroneous and non-dependent) ***pieces of knowledge, th ere is an aspect in which a specific knowledge is more accurate than another, while that another is more accurate in another aspect.*** *The conjunction of pieces of knowledge—being knowledge, too—is at least as accurate as the other pieces of knowledge.*)

This allows for defining the **information volume of a system's representation** and the **information volume of knowledge** in an aspect as the accuracy of the knowledge (approximation) belonging to that representation in that aspect.

(The information volume of a representation or knowledge, provided that it is not the arbitrary abstract system, is information regarding the initial system.)

Knowledge Acquisition

Consider a system and its consequences $\sigma_{i\in I}$ (where *I* is an ordered set of indices with the minimum *i0*) such that any "subsequent" system (systems with a greater index) is a comparable specialization of its predecessors or, in other words, where systems do not diverge from the initial system. Such a series is called the **monotonic approximation** of the system. If there is at least one more accurate consequence than the initial element σ_{i0}, the approximation is called **contraction**.

Knowledge acquisition about or **learning** a given system σ from an aspect $\prec$ is defined as a series of systems treated pieces of knowledge such that the conjunction of the accuracies (called the **accuracies of learning**) of those systems in the series form a contraction in the studied aspect.

Only the accuracies of pieces of knowledge have to form contraction for knowledge acquisition. That is, the process may contain knowledge diverging from the approximative system (such as independent or contradictory knowledge), since only the relations of the accuracies of the knowledge pieces are of relevance. Learning, for example, may include the generation of a piece of knowledge dependent on the system once its accuracy differs less than that of the initial knowledge.

For this reason, a contraction is an aspect-independent "ideal knowledge acquisition" (or from another point of view, an idealized way of acquiring knowledge) that never diverges from the initial system, but generates a more or less accurate abstract image of it.

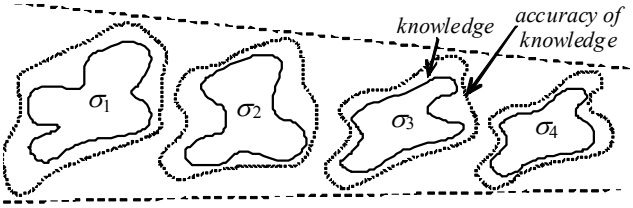


Figure 7: Learning determines more accurate boundaries of the system

(Taking temperature values as an example, the knowledge of the temperature range of [22C°;25C°] following the initial range of [20C°;24C°] becomes knowledge acquisition regarding the actual range of [22C°;23C°]. Here [20C°;24C°] is the initial accuracy, and the accuracy of the subsequent measurement is [22C°;25C°], the conjunction of which with the preceding knowledge is [22C°;24C°], therefore the repeated conjunctions of accuracies form contraction.)

Since the series of systems defined on an ordered set is a process, too, *knowledge acquisition may be studied as a system*.

Example-Driven Knowledge Acquisition

Investigated systems usually could not be cognized in one step. Approximations of complex systems may only be specified by connecting details treated atomic, closed and “known”, by connecting or relating their elements using our partial knowledge and comprehension. Since following “known” methods when building upon “known” initial systems, the resulted system will also be "known". Therefore, once our knowledge changes equatably to the change of the system to approximate, we have cognized the desired system. Putting together, *knowledge acquisition is a representation method*, too.

For knowledge acquisition we have to test knowledge to find out how well it fits the system and how much it differs from the system to be approximated (the two operations are collectively called *comparison*), while, however, we also have to *modify* the knowledge based on the discovered divergences. Let’s consider the following procedure as the knowledge acquisition process (cognizance) of a system σ: Take an arbitrary piece of knowledge such as the arbitrary abstract system $\aleph$. If the knowledge is *abstract* too much (because it is a consequence of the approximated system), perform a comparable specialization on it. If the knowledge is accurate too much (because it is only an *example* of the approximated system without including all the possible examples), perform a comparable abstraction on it, practically one that includes the missing examples. If the systems are independent, perform such a comparable abstraction that yields, apart from knowledge, a comparable abstraction of the approximated system. Should the piece of acquired knowledge be exact, i.e., equatable to the approximated system, then we have cognized the system.

For this reason, a comparable abstraction or comparable specialization is just enough to take one step in modifying our knowledge. The comparable abstraction may be specified by a disjunction with an (appropriate) complementary system. The comparable specialization, on the other hand, may be specified by a conjunction with an (appropriate) complementary system. Third, if an example included in the knowledge is not part of the system, the system may be further specialized by a conjunction with the converse of the example. Therefore, the apparatus necessary to modify

knowledge can be reduced to using conjunction and disjunction on the current knowledge and a complement.

More accurate knowledge can be acquired by repeated modifications based on searching for missing or unnecessary examples in the approximation. Therefore, this process is *knowledge acquisition*, a learning process, called **example-driven knowledge acquisition**.

Analysis and Synthesis

Analysis, in our principle, refers to the act when a bigger unit is described as a given kind of composition of some given initial elements. Two major terms have to be defined for this approach: what we build upon (called the concepts of the system) and the way we build upon (or way of composition, called method).

Shifting a system from its own time and location to another time and location with a renotation is called **direct positioning**. Renotations relocating a system to the time and locations of another system are called **positioning**. The **actual concept of a system** is defined as the consequence of its constraint, while the direct positionings of actual concepts are the **possible concepts of the system**.

A **concept of a system** is an appropriate positioning of one of its actual concepts. Such a positioning is called the **application of the concept**. (Application answers the question: "Where and when does it exist in that system?")

(Actual concepts are possible concepts, too. Consequences, examples and composite components of a system are also actual concepts. Actual and possible concepts are concepts of the system. *Possible concepts of an instance* (and, hence, probably the possible concepts of reality) *are actual concepts and therefore consequences of the system.)*

Repetitions of the behavior of a system occurring in other moments or locations may be extracted or excerpted to form a possible concept of the system. The actual moments and locations of that possible concept may be specified by its direct positioning.

The **actual domain of a concept** of a system is the set of disjunctions of the system's actual concepts which, when positioned appropriately, result in a system equatable to the concept ("the concept really gets manifested there"). The **possible domain of a concept** is the set of the disjunctions of

the possible concepts of the system which, when positioned appropriately, result in a system equatable to the concept ("the concept could appear in this or that form there").

(The possible domain of a given concept of a system is the consequence of its actual domain.)

An "actual concept" of a system is usually a concept more extensive than the information regarding the system. Non-trivial examples of non-instance systems are actual concepts, but are not information.

(Actual concepts of an instance are utmost comparable abstractions of that instance, since instances may not be constrained further, that is, all but the arbitrarity ($\aleph$) are information regarding the instance.)

But how can greater systems be built from concepts?

We can define the product of comparable systems using a given set of indices (where indices are called **roles**), where a subset of the product becomes **relation between the systems**. (For example, in case of the role set (husband, wife, child), an element of the product may be the following: (husband: *James*, wife: *Jane*, child: *Jimmy*, child: *Nancy*). The relation of similar products—all relation of the same kind of correlation—can be called family.) The function assigning a comparable system to a subset of products is called a **method of comparable systems**.

(Conjunctions, disjunctions and negations of a system are methods. Methods can be rewritten to relations.)

Now we have the apparatus for complex specifications, since we have answered the question of "from what" (concepts) and the "how" (along methods). Therefore, specifying (or defining) a system as a method is called **specification as macrosystem**, and the concepts (being in the role of parameters) are called **building elements** or **constituents** of the macrosystem. A method may be, of course, a complex method built from several more atomic methods.

Specifying a system as a macrosystem is called the **analysis** of the initial system and, simultaneously, the **synthesis** of constituents.

Possible and Actual Components

Earlier stated, a piece of initial knowledge (considered an approximation) can be specialized (based on the comparison of the tests performed on the

examples) to be more accurate by extending it towards abstraction and specialization. Generating macrosystems based on extending or narrowing the possible state combinations precisely follows the process made of steps of approximations. For this reason, this method *specifies an approximation of a given system along with capturing the knowledge acquisition process.* From the opposite point of view, **a macrosystem, in addition to specifying a piece of knowledge, also specifies the learning process or, in other words, presents (or interprets) knowledge using concepts assumed to be known.**

Byways of a given knowledge acquisition process and the particular results of given steps abstracted later can be rearranged into a "*more ideal*" knowledge acquisition process. This kind of specification therefore describes an **abstract information acquisition process** with *fewer byways, less pieces of non-abstracted particular knowledge, and with the possibility to interchange certain steps*.

Specializing and abstracting extensions performed using complementary systems can be rewritten to conjunction, disjunction and negation operations which, in turn, can be transformed (by correlating components or by "layers"), and certain steps can be *aggregated*, resulting in a "more ideal" knowledge acquisition process containing less byways and particular results. A possible result of aggregation is the **disjunctive normal form** made of disjunctions of conjunctions, or the **conjunctive normal form** made of conjunctions of disjunctions.

A consequence of a system is called **actual component**, while the condition (comparable specialization) of a component is called **possible component**.

(Systems included in a given system as possible or actual components are concepts of that system. Components of a given system are possible and, simultaneously, actual components of the system.)

(When performing disjunction on some given initial systems, those systems are possible components of the results system. Initial systems of a conjunction are actual components of the result system.)

III.

Rationality and Analogy

Model

Our principle distinguishes the **abstract model of a system**, defined simply as a type abstraction (a function-like relation) of a system, from the **model of a system**, being an instance or a given particular variant of the abstract model. (A model is, therefore, a simple representation, while an abstract model is an abstract representation.)

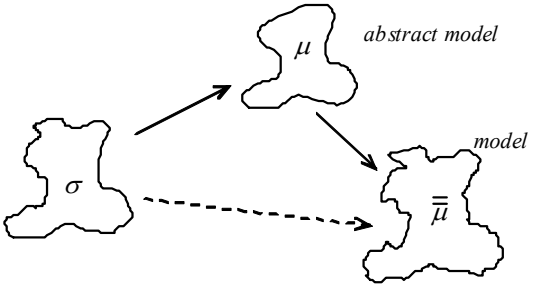


Figure 8: Abstract model and model.

(The accuracy of an abstract model is the abstract model itself, since it is an approximative or exact representation. A model's accuracy, however, is the abstract model of the model.)

(The arbitrary abstract system is an abstract model of every system.)

(Think of a man touching a post when passing by and getting closer to it by tensing its muscles, thus reaching higher speed. This man has just constructed the model of a satellite gaining speed when passing a planet. The essence behind the model—its semantics—is the abstract model describing, using the laws of Physics, the increase in circumferential speed of a body forced into circular motion by an effort greater than necessary for centripetal acceleration.)

Type abstraction is a "gentle" abstraction, since it describes a function-like relation between a system and its type, and therefore its representation as a model is also a "gentle", an easy-to-understand representation.

(A composition performed on the type abstractions of the components of a composite becomes an abstract model of the initial system, so the model of the composite may be constructed from the models of its constituents.)

Quantum System, Quantum Change

In case of a general system, the set of moments is an arbitrary ordered set, such as the continuous set of real numbers. Often, however, it is enough to study changes only in some given moments, called "milestones" and consider changes to be arbitrary in the intervals between the milestone moments. The change, originally continuous, then becomes a series of snapshots, or a set of transitions resembling jumps, that is, "quantum jumps" of states.

An ordered set is an **ordered quantum set**, if the **successor**, once exists, is known for any element. A **quantum set** is a set for which such an ordering can be defined that yields an ordered quantum set.

(An ordered set is an ordered quantum set if and only if a subset of integers can be assigned to it via strongly monotonic one-to-one (bijective) correspondence. The cardinality of a quantum set is utmost countable.)

A system is called **quantum system** if the set of moments (in its time set) in which the function is defined is an ordered quantum set. Systems studied only in quantum moments are called **quantized systems**. The image captured by quantizing a system simplifies both studying and representing changes. (A quantized system is an abstract model of the initial system and, additionally, a quantum system.)

Should a minimum exist for the set of moments for a system in which the system is defined, that moment is called the **initial point** or **starting point**. (The change of a quantum system with existing starting point can be represented as a series of states $s_0 \mapsto s_1 \mapsto s_2 \mapsto \dots$ (the successor state of s_0 is s_1, while its successor is s_2 and so on), that, in finite cases, turns to be $s_0 \mapsto s_1 \mapsto s_2 \mapsto \dots \mapsto s_n$).

(Defined changes of a quantum system after a given moment can be specified as a series of states.)

If a system is studied only in two moments, the resulting system is actually the least unit of change and is called the **state change** or **step** of the system. Steps may be treated as state transformations assigning state *s* in the defined moment *t* a state *s'* in a moment *t'*.

Knowledge Acquisition as a System

Knowledge acquisition itself is also a process, i.e., a system, where the ordered set I is considered as the set of moments, and pieces of knowledge are the states.

(Accuracies of a piece of knowledge acquisition form knowledge acquisition.)

Assigning "moments" *i* of knowledge acquisition the accuracies $p_i^{(\prec)}$ of aspect $\prec$ (those being considered the results of the knowledge acquisition), the result system provides an abstract description of the knowledge acquisition process, and therefore is called **abstract knowledge acquisition process**.

(All abstract knowledge acquisition processes are contractions, and all contractions are abstract knowledge acquisition processes. Contraction, for this reason, as an "ideal" way of knowledge acquisition, specifies knowledge acquisition only to the extent of its accuracies (to be considered the results of that contraction), that is, a single contraction specifies a set of several knowledge acquisitions, or *abstract knowledge acquisition*.)

A monotonic approximation $\sigma_{i\in I}$ of a system σ is called the **strongly monotonic approximation of** σ if it approximates the system with more and more accurate systems, that is, if system σ_i and system σ_j are inequatable systems for $\forall i < j$ $(i, j \in I)$.

(A strongly monotonic approximation is contraction if it contains at least two elements.)

Taking a system σ, a series of pieces of knowledge $\sigma_{i\in N}$ comparable to the system (where *N* is the set of natural numbers or a subset 0…*n* of it) is the **stepwise approximation** of the system.

Quantizing a knowledge acquisition process means that the process is studied only at specific milestones, with all intermediary states left ignored. *Thus quantizing simplifies representation of knowledge acquisitions.*

(A quantized knowledge acquisition processes can be represented as a series. *A quantized abstract knowledge acquisition process can be represented as a series of accuracies.)*

(Quantized knowledge acquisition processes are stepwise approximations.)

When quantizing knowledge acquisition, a subsequent milestone may be a new piece of knowledge σ_i featuring some novelty, that is, a piece of knowledge the accuracy of which is not the consequence of current accuracies noted by $p_i^{(\prec)}$.

(A contraction can be quantized to a strongly monotonic approximation having at least two elements, which itself is an "ideal" knowledge acquisition process, too.)

Intermediary knowledge not featuring any novelty may be ignored or skipped by careful selection of milestones, so this approach yields in a strongly monotonic approximation. This produces "more ideal" knowledge acquisition that is easier to represent.

Approximation by Stepwise Finite Selections

Our approach often employs the tool of *selection*, which is the specification of a given subset of a set. Even the definition of system is based on selections, since the set of possible states in various moments is specified as some subsets of the complete set of all states representing completeness. Comparable specialization also relies on selections due to the need of narrowing down the possible state combinations tied to given moments or, in other words, subsets of certain sets have to be determined. Knowledge acquisition has to specify a system more accurate than the knowledge at the

starting point, therefore selection also plays fundamental role in the knowledge acquisition process. Selection usually means a quantized operation (assigning a set one of its subsets), however the term of "continuously progressing selection" (or continuous selection, for a simpler form) may make sense, too.

The selection method can be further constrained if only a *finite number of alternatives* are allowed to be the choices of the selections in the various steps. Such a method can be further decomposed to yes/no decisions. The method of stepwise selections (making selections in each step) implies a specific representation method, a representation by a series generated from a finite set of symbols, called **description**.

Let's perform the approximation of a studied *deterministically specified* system as follows. Let's classify the possible states of the system into some abstract states ("some" hereafter used interchangeably with "finite number of"), then by selecting a moment of the system's time, determine what class corresponds to the abstract state the system rests in. After this, either take another moment of the system's time and determine the abstract state class in that moment, or specialize the state in the earlier selected moment by decomposing the abstract state class into some (i.e., a finite number of) subclasses. The only restriction is for the class hierarchy of the abstract states, which needs to be common for the entire system or, in other words, no class can have more than one subclass-decomposition. The systems obtained by performing the steps of the method approximate the studied system better and better. The sets of moments and states in which the system is defined are finite sets for all systems, since we have decomposed a finite number of classes into a finite number of subclasses during a finite number of steps. Such a series of systems is then called an **approximation by stepwise finite selections** of the studied system. The result of each step is a type specialization of the preceding systems, while it is also a type abstraction or abstract model of the studied system. The state set of such an approximative system contains the subclasses of the lowest level of the class hierarchy. If state were determined not this precisely in a given moment, all the lowest-level subclasses of the state class have to be listed as possible states, following an indeterministic specification.

A contraction $\sigma_{i\in I}$ of system σ is an **arbitrarily accurate approximation** of σ if σ_i and σ_j are inequatable systems (σ_i is a non-trivial condition of σ_j) or σ_j and σ are equatable systems for $\forall i < j$. Arbitrary accuracy is used in the sense that all pieces of knowledge subsequent to a given piece of inexact knowledge will be (a non-trivially) more accurate approximation for all inexact pieces of knowledge.

(Approximation by stepwise finite selections is an arbitrarily accurate approximation.)

(Approximation by stepwise finite selections is a contraction and for that reason, it is also a knowledge acquisition process.)

Symbolic Model

A way to perform the mentioned selections and, therefore, a way to represent knowledge is to take a finite set of symbols (containing the appropriate number of elements) for each selection and then to assign each possibility (subclass) a symbol and, finally, to denote each selection with a symbol. This way knowledge can be represented by a series of value pairs of the selected symbol and the symbol set (as value and type). Another approach to representation is to assign a finite series of symbols taken from a (finite) set of base symbols (with the series being the "names" or identifiers used as indices in the selections) to each alternative in the type hierarchy.

(Note that it holds for systems of both deterministic and general specification that the cardinality of the defined state combinations may not exceed the cardinality of defined moments.)

A system is a **finite or bounded system** if its time is finite (implying that the set of defined state combinations is finite, too).

(Approximation by stepwise finite selections produces a finite abstract model of the system in each step.)

A **finite or bounded symbolic model of a system** $\mathcal{M}^{\Sigma}\sigma$ is the (finite) abstract model $\mu = (T^{\Sigma}, S^{\Sigma}, P^{\mu})$ of the system, where T^{Σ} and S^{Σ} are finite symbol sets and, moreover, T^{Σ} is ordered.

(Finite symbolic models of a system are type abstractions of that system.)

(A strongly monotonic assignment of a finite subset of the natural numbers to a finite subset of the defined moments of a system, and the assignment of a finite subset of the natural numbers to the state set of that system together yields a finite symbolic model of the system, where the two subsets of the natural numbers play the role of the two symbol sets.)

Finite abstract models, from a specific aspect, are disclosed, since the specification of an abstract model defined by n number of moments and m state classes may not evidently be represented by a model defined by the same number of moments and state classes. A more accurate approximation of systems, called the *rational symbolic model*, assigns not only an element (a one-element series) of a finite set of symbols to the moments, but may rather assign a finite series, or may also denote the moments of the system with finite, ordered series of symbols. Series of a finite set of symbols with finite (but otherwise arbitrary) lengths may even form an infinite set, enabling the rational system to be defined at an infinite number of moments and to have an infinite number of actual states, but still, each moment and the state belonging to that moment may be determined by a finite number of steps of an arbitrary stepwise approximation by finite alternatives.

Quantized abstract models take place between the finite and the rational models, and allows for assigning a finite series of symbols to each moment of the system, while requiring the "successor" to be able to be defined for each moment but the last. Quantum systems therefore may be called **integral systems**, too, while quantum systems based on a starting point may also be called **natural systems**.

(Finite systems are natural systems. Finite and natural systems are both integral systems.)

(The set of all quantized abstract models of a system is identical to the set of all integral abstract models of that system, and the set of all quantized models is identical to the set of all integral models.)

A system is a **rational system** if it contains only a countable number of defined moments (implying that the set of defined state combinations is countable, too).

(Finite, natural and integral systems are rational systems, conjointly.)

(The result of specializing a rational abstract model by an approximation by stepwise finite selections is also a rational abstract model.)

A (rational) abstract model $\mu = (T^{\Sigma^*}, S^{\Sigma^*}, P^{\mu})$ of a system is a **rational symbolic model** ($\mathcal{M}^{\Sigma^*}\sigma$) if T^{Σ^*} and S^{Σ^*} are the sets of series of symbols produced from the appropriate finite symbol sets T^{Σ} and S^{Σ}, and T^{Σ^*} is ordered. A rational symbolic model of a system is an **integral symbolic model** if it is also a quantum system. The **natural symbolic model of a system** is its startpoint-based integral symbolic model—that is, an integral symbolic model is natural symbolic model if the set of defined moments have a minimum value.

(A strongly monotonic assignment of a subset of the natural numbers to a subset of the defined moments of a system together with an assignment of a subset of the natural numbers to the state set of that system yields in the natural symbolic model of the system.)

(A strongly monotonic assignment of a subset of integer numbers to a subset of the defined moments of a system together with an assignment of a subset of integers to the state set of that system yields in the integral symbolic model of the system.)

(A strongly monotonic assignment of a subset of rational numbers to an utmost countable subset of the defined moments of a system together with an assignment of a subset of rational numbers to the state set of that system yields in the rational symbolic model of the system.)

Rational abstract models are also disclosed from a certain point of view. Although they can represent the results of any given step in an arbitrary stepwise finite specialization, they are not capable to represent the result of an infinite number of specializations or, in other words, they cannot represent the limit of approximations extending beyond all bounds. *Real symbolic models*, enabling more accurate representations of systems, can assign arbitrary, even infinite series of symbols to moments and states.

A system is called a ***real system*** if its time is utmost of continuum cardinality (yielding that the cardinality of defined state combinations is utmost continuum, too).

(Finite, natural, integral and rational systems are also real systems.)

(Applying the method of approximation by stepwise finite selections an arbitrary, even infinite number of times generates the real abstract model of a system.)

A (rational) abstract model $\mu = (T^{\Sigma^{**}}, S^{\Sigma^{**}}, P^{\mu})$ of a system is a **real symbolic model** ($\mathcal{M}^{\Sigma^{**}}\sigma$ or $\mathcal{M}^{\Sigma^{\infty}}\sigma$), if $T^{\Sigma^{**}}$ and $S^{\Sigma^{**}}$ are sets of series of symbols of arbitrary (even infinite) lengths, produced from the appropriate finite sets of symbols T^{Σ} and S^{Σ}, and $T^{\Sigma^{**}}$ is ordered.

(A strongly monotonic assignment of a subset of the real numbers to a subset of defined moments of utmost continuum cardinality, together with an assignment of a subset of the real numbers to the state set of the same system, yields in the real symbolic model of the system.)

Stepwise finite selection can be reduced to simple yes/no decisions. The approach of discrete selections fundamentally determines the scope of information that can be acquired about a system, and specifies three quality levels depending on the number of decisions. Limiting the number of decisions by a pre-defined value results in a bounded or finite abstract model of the studied system. Enabling an arbitrary but finite number of decisions yields in the rational abstract model, while enabling arbitrary, even infinite number of decisions ends up in the real model.

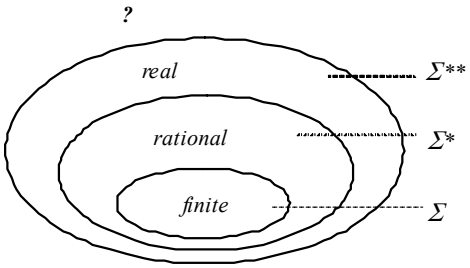


Figure 9: Bounded, rational and real abstract models.

Approximation by stepwise finite selections is capable to model values. Values are actually indices identifying actually selected alternatives amongst all possible alternatives. Values can form a hierarchy (such as for “green”

one can define “harsh green” and “very harsh green” as sub-alternatives), therefore accuracy may be defined for those. On the other hand, values may be complex (in case of different aspects; in case of a car, for instance, one can define “fast, green, comfortable”).

Rationality and Analogy

A direct consequence of the method of *stepwise finite selection* (or stepwise selection from a finite number of alternatives) are the rational image and the real image of the initial system, the latter being the abstracted bound of the rational image. Is there an approach beyond the real one enabling acquisition of a greater scope of information and fastening the approximation of the studied system? For a quality change, selection from a set of alternatives with (practically and potentially) *infinite*, even continuum cardinality has to be allowed in each step. But how to imagine such an analogical knowledge acquisition? Consider each state of the system to cognize (learn) as a complex wave. The resonance capable to represent the major curves of the waveform is a selection from infinite alternatives. Analogical knowledge acquisition, this way, can be imagined as “tuning”, as a monotonic approximation by a method based on the principle of resonance. Representation, furthermore, can be imagined as something that generates a complex waveform during interpretation—in case of a poem, for example, vowel gradations, the forms related to its words and phrases generate a complex “waveform” when reading. Analogical approximation enables for faster acquisition of a greater scope of knowledge, but the complex knowledge acquired this way can not be verified completely with rational apparatus, but only with an analogical one.

In spite of its bounds, rational approximation features more advantages. Notwithstanding its slowness, it is capable for *arbitrarily accurate approximation* of systems, while, on the other hand, approximations are easy to track and document and, therefore, easy to be verified.

IV.

Impact and Relationship

Abstractly Specified Systems

The systems mentioned earlier all described changes happening a certain way, however those changes could be abstract to some extent and could have some uncertainty in its states taken at the various moments. The "changing change" defined below reflects various behaviors.

In the **abstract specification** of the form $\sigma^{(\mathcal{A})} = (\mathcal{T}, S, P^{(\mathcal{A})})$ of (direct comparable) systems, $P^{(\mathcal{A})}$ assigns each argument $a \in \mathcal{A}$ a change $P^{(\mathcal{A})(a)} : T \to 2^S \setminus \{\varnothing\}$ (or for short: $P^{(a)}$) called the **run** of **behavior** of the abstractly specified system. Abstract specification is suitable to define "changing change", that is, when the way a certain change happens depends on an argument.

Those arguments may form an arbitrary set and, in general case, may involve elements "external" to the world of the system. The arguments, however, may also be certain state combinations of the system, such as the state combination s_0 taken in a designated moment t_0. The result in this case is a function of the form $P^{(S_0)}$ determining, for each moment *t,* the actual state combination $s_t^{(s_0)} = P^{(S_0)(s_0)}(t)$ of the system with the state combination s_0 taken in moment t_0 (S_0 is the set of possible states in moment t_0). Such abstractly specified systems are called **transition functions**.

(*Any abstractly specified system can be defined by a transition function* simply by adding the argument to the system as a parameter-like component. This also can be viewed as placing the system into the context of the parameter-like component.)

(An (abstractly specified) system defined by a transition function can be defined as a unary method.)

Certain startpoint-based quantum systems can be defined by a function of the form $P^{(\rightarrow)} : \overline{S} \rightarrow \overline{S}$ by parameterizing the function with a state combination $s_{t_0} \subseteq \overline{S}$ belonging to a given moment t_0, where the function $P^{(\rightarrow)} : \overline{S} \rightarrow \overline{S}$ determines the state combination s' of the subsequent (successor) moment for each state combination *s*, provided that such a moment exists. Such systems are called **pure** (or **regular**) **transition functions**. The system is transitioned from the initial state combination *s* to state combinations $s' = P^{(\rightarrow)}(s)$, $s'' = P^{(\rightarrow)}(s') = P^{(\rightarrow)}(P^{(\rightarrow)}(s))$, ..., respectively.

(Pure transition functions are transition functions.)

Behavioral Conditions and Constraints

A certain set of arguments $\mathcal{C} \subseteq \mathcal{A}$ of an abstractly specified system $\sigma^{(\mathcal{A})}$ form the **conditions of behaviors** or the **condition arguments** of $\sigma^{(c)}$ ($c \in \mathcal{C}$), while runs $\sigma^{(c)}$ ($c \in \mathcal{C}$) are **behaviors of condition** $\mathcal{C}$. (Since being an abstractly specified system, denotation $\sigma^{(\mathcal{C} \subseteq \mathcal{A})}$ or its abbreviated form $\sigma^{(\mathcal{C})}$ can be used.)

Upon merging behaviors $\sigma^{(\mathcal{C} \subseteq \mathcal{A})}$, the union of those of the form $\bigcup_{c \in \mathcal{C}} \sigma^{(c)}$ is the **abstract behavior of condition** $\mathcal{C}$ (written as $\bigcup \sigma^{(\mathcal{C})}$) or **conditional abstract behavior** of the system. This is actually the boundary or "bed" of runs of condition $\mathcal{C}$, i.e., behaviors in case of arguments $\mathcal{C}$ stay "within" this abstract system. If the condition is the complete set of arguments, the result is the union $\bigcup \sigma^{(\mathcal{A})}$ of all runs, called the **unconditional abstract behavior**.

A certain σ' restricting all behaviors of a $\sigma^{(\mathcal{C} \subseteq \mathcal{A})}$ is called the (conditional or unconditional) **constraint of the behaviors** or, in other words, **constrainer condition of behaviors**, since σ' sorts out all the behaviors for which it is satisfied. The **consequence of behaviors**

($\sigma' \Leftarrow \sigma^{(C \subseteq A)}$) may be defined similarly, if σ' is the consequence of each behavior. (Any consequence of some given behaviors is either the constraint of those behaviors or it is the arbitrary abstract system itself.)

To summarize, we can sort out some given behaviors (of a given condition) by a subset of the arguments (the condition arguments). The common "bed" of the selected behaviors specify an abstract behavior with a given condition. Given behaviors, on the other hand, may be sorted out by specifying a system as a constrainer condition, i.e., by specifying the constraining condition of the behaviors. The two terms are closely coupled.

(σ' *is the consequence of behaviors of some certain conditions if and only if it is the consequence of the abstract behavior of that condition*; formally when $(\sigma' \Leftarrow \sigma^{(C \subseteq A)}) \Leftrightarrow (\sigma' \Leftarrow \bigcup \sigma^{(C \subseteq A)})$.)

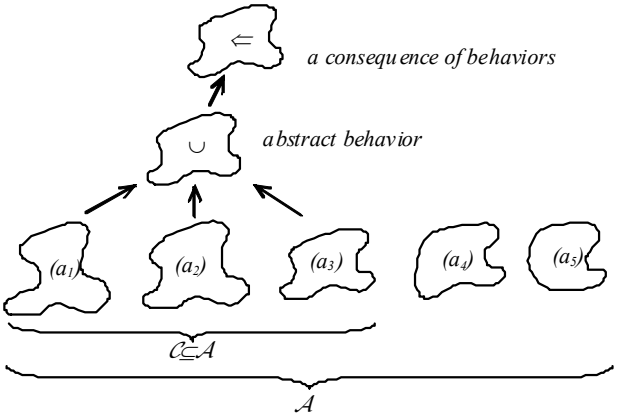


Figure 10: Domain of behaviors and condition arguments of a consequence.

A system comparable by behaviors (called the knowledge regarding the behaviors) generally sorts out behaviors for which it applies as a consequence (comparable abstraction). The subset $\mathcal{C}_{\sigma'} = \{c \in \mathcal{A} \mid \sigma' \Leftarrow \sigma^{(c)}\}$ of the arguments of the behaviors becomes a condition (set of condition arguments), therefore it is called the **condition of the knowledge regarding the behaviors**.

(The condition of the knowledge σ' regarding the behaviors is an empty set if and only if σ' is not the consequence of any of the behaviors. The

condition of knowledge σ' regarding the behaviors is the complete argument set $\mathcal{A}$ if and only if σ' is the unconditional consequence of those behaviors. The behaviorally comparable arbitrary abstract system is the unconditional consequence of the behaviors.)

Entity Constrainers

The term **entity** is used hereafter for a given part of a composite (i.e. for state changes at locations π_E). Since a composite is a component of itself, terms and statements regarding entities may be automatically applied to the complete system as well.

Narrowing down the scope of studying behaviors $\sigma^{(\mathcal{C})}$ to some selected moments $T' \subseteq \mathcal{T}$ and an entity E can be described as $E^{(\mathcal{C})}_{|T'} = \sigma^{(\mathcal{C})}_{|E|T'}$. Abstract behavior of entity E, written as $\bigcup E^{(\mathcal{C})}_{|T'}$, is called the **domain of behaviors** $E^{(\mathcal{C})}_{|T'}$ of entity E. *(The domain extends in case of a less restrictive condition, i.e., for* $\mathcal{C}' \subseteq \mathcal{C}$, $\bigcup E^{(\mathcal{C}')}_{|T'} \subseteq \bigcup E^{(\mathcal{C})}_{|T'}$. Since the entity may be more diverse in case of a greater set of arguments, its abstract behavior, as well as its domain of behaviors may be extended.

A constrainer of a system is a **functional constraint** or, in other words, **value constrainer** if it is an instance, that is, it narrows down the set of possible states to a single value. Non-value constrainers (those constraints enabling for multiple possible values) are called **relational constrainers**, **range constrainers** or **domain constrainers**.

When studying an entity with behavior $E^{(\mathcal{C})}$ in selected moments $T' \subseteq \mathcal{T}$, it turns out that the **function of change** (or **change function**) **constrains the behavior of the entity** in those moments (noted as $\rightarrow^{(\mathcal{C})}_{T'} E$), if $\bigcup E^{(\mathcal{C})}_{|T'}$ is not the arbitrary abstract system (but a constrainer of E), otherwise the entity is an **unconstrained entity** in those moments (written as $\not\rightarrow^{(\mathcal{C})}_{T'} E$). Entity constrainers may be value constrainers, as well as domain constrainers.

Constraining a specific behavior of an entity means that the set of state combinations taken by the entity at locations π_E in the given moments may be narrowed down.

(An entity constrainer is the constrainer of the given behaviors of the entire system, that is, an entity constrainer is *information* regarding behavior.)

Properties held true in any defined moments of a system are called **structural properties**. Otherwise, that is, when there are defined moments in which the property is true, and others when it is violated, that property is called a **temporary property** (or **temporal property**). In the notation of structural properties, indication of the set of all defined moments can be safely ignored (such as for a structural constraint of an entity one can write $\rightarrow^{(C)} E$).

Given the constraints or consequences of an entity's behaviors, we can tell which one is stronger or weaker—in a given aspect, of course. If σ' is just not a behavioral consequence of the abstraction $\prec_{(x)} (\sigma^{(c)})$ of the behavior in an aspect $\prec$, but another one is still a consequence, then σ' is the **stronger consequence**.

Considering two dependent consequences, the "stronger" always specifies a smaller set for possible states, while "weaker" ones enable more such states. (The arbitrary abstract behavior is weaker than any other consequences. *The comparable abstraction of the consequences of behaviors is a weaker consequence of the behavior.*)

(*Abstraction weakens constrainers* meaning that only a weaker consequence may belong to the comparable abstractions of behaviors than the consequence of the given behaviors.)

Impact, Dependency and Relationship

The impact between components or entities of a system may be defined as follows: the impacting entity constrains the changes of the dependent entity (or "channels those changes into a bed"), therefore changes of the impacting may force change in the dependent, if this latter is not still in the appropriate "bed" according to the constraint. Impact is a conditional (*if-then*) constraint:

"if" the impacting entity shows a given behavior, *"then"* some otherwise possible behaviors are not enabled for the dependent entity.

Taking behaviors of a condition $\mathcal{C}$, the **impact of** (a given) **behavior** $A^{(c)}$ **of an entity** A **on an entity** B in certain moments $T' \subseteq \mathcal{T}$ is defined as follows: Let's collect those behaviors among all of a condition $\mathcal{C}$, in which entity A changes the same way as behavior $A^{(c)}$ does at the selected moments, and map the union of changes with the same condition in B (noted: $B_{(\leftarrow^{(\mathcal{C})}_{T'} A^{(c)})}$) to the behaviors selected. This is what entity B can be like when entity A changes the same way as $A^{(c)}$ does. This can be compared to the domain of changes $\bigcup B^{(\mathcal{C})}_{|T'}$ of entity B with condition $\mathcal{C}$, and if it is narrower (i.e., $B_{(\leftarrow^{(\mathcal{C})}_{T'} A^{(c)})} \subset \bigcup B^{(\mathcal{C})}_{|T'}$), then the (specific) behavior $A^{(c)}$ of entity A in moments T' **impacts** behaviors $B^{(\mathcal{C})}$ of the dependent entity (noted as $A^{(c)} \rightarrow^{(\mathcal{C})}_{T'} B$, while in the opposite case it is noted as $A^{(c)} \not\rightarrow^{(\mathcal{C})}_{T'} B$). Impact of multiple behaviors $A^{(\mathcal{C}')}$ can be defined similarly.

(Impacts may disappear and appear by extending the condition.)

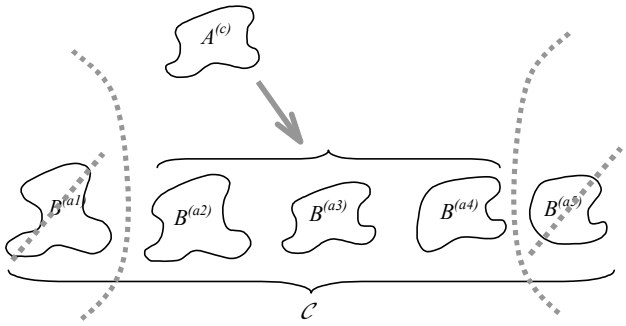


Figure 11: Impact "channels" the change of the dependent into a "bed".

Impact and dependency can be defined in general cases as follows: Should a (specific) behavior $A^{(c)} \in A^{(\mathcal{C})}$ with a given condition $\mathcal{C}$ of entity A impact behaviors $B^{(\mathcal{C})}$ in some studied moments, then we say entity A

impacts behaviors $B^{(\mathcal{C})}$ of entity B (noted: $A \rightarrow_{T'}^{(\mathcal{C})} B$), and behaviors $B^{(\mathcal{C})}$ **depend on** entity A (noted as $B \leftarrow_{T'}^{(\mathcal{C})} A$).

Mutual impacts ($A \leftarrow_{T'}^{(\mathcal{C})} B$ and $A \rightarrow_{T'}^{(\mathcal{C})} B$) occurring at some given moments and in case of a condition are called **interaction** (noted as $A \leftrightarrow_{T'}^{(\mathcal{C})} B$). By ignoring the direction of the effect, that is, considering only that one entity impacts another ($A \leftarrow_{T'}^{(\mathcal{C})} B$ or $A \rightarrow_{T'}^{(\mathcal{C})} B$), then the two entities are in **relationship** with each other ($A \sim_{T'}^{(\mathcal{C})} B$). If there are no impacts in any directions, the two entities are said to be **independent** of each other in the studied moments (noted: $A \not\sim_{T'}^{(\mathcal{C})} B$).

(Impacts between entities are transitive, therefore the impact of an entity may extend to several other entities.) Relationships, however, are not necessarily transitive. For example, in case of $A \rightarrow B$ and $A \rightarrow C$, B and C may be independent.)

(An entity of invariant behaviors does not impact other entities and is not dependent on other entities, therefore it does not impact itself, and neither is dependent on itself. The arbitrary abstract system does not impact any entity.)

Impact Strength

Impact itself acts as a constraint, too, therefore, based on our earlier definitions, the aspect-aware weakness and strength of *impacts as constrainers* can be interpreted.

(*Abstraction weakens impacts as constraints.* Weaker consequences can be constructed against the comparable abstractions of behaviors, meaning that only weaker impacts can be specified as constraints.

However, a strength and weakness of another nature can also be defined between impacts. Only weaker constrainers can be specified for more abstract behaviors, so impacts dissipate during subsequent abstractions of a system, yielding an increase in independencies. An **impact** and **relationship** is said **stronger** in an aspect, if it dissipates only at a higher-level abstraction of that aspect.

(*Impacts do not necessarily apply for the consequences of the behaviors.* First, the abstractions of the various behaviors of the impacting entity may become equal, causing one of its behaviors to become identical with another that no more impacts the dependent entity. Considering the abstractions of the behaviors on the other hand, behaviors of the dependent entity may also become more abstract, leading to the weakening of the impact (as a constrainer).)

(***Impacts may appear in consequences of behaviors.*** Think of a glass held in a hand. Should you release the glass and say "Fall down", the glass will fall down, however it were fallen down even if you hadn't say a word. An abstraction here causes that such magic words seem to be necessary for the glass to fall down. Those impacts in the consequences of the behaviors, which were not present in the initial system are called **phantom impacts** (**appearing in a consequence**).

A comparable abstraction is called the **domain abstraction of behaviors** if the domain of behaviors stays the same for all entity E, that is, $\bigcup E^{(C)} = \bigcup E'^{(C)}$.

(New ("phantom") impacts may not appear in domain abstractions. Domain abstractions cause increase in independencies.)

Entity, Context and External World

We can choose to extract and study a portion, an entity of a system. This way the composite can be decomposed to an *entity* and the world around that entity: $\sigma = E \otimes E_{(\varnothing)}$. Those components of the world around the entity, which impact entity E are called the entity's **input** (notation: $E_{(\leftarrow)}$), while those impacted by the given entity are called the entity's **output** (notation: $E_{(\rightarrow)}$). Those components of the system which *simultaneously* impact and depend on the entity (that is, which appear both in its input and output) are the **transfers** of the entity, noted $E_{(\leftrightarrow)}$. The **context** of an entity ($E_{(\sim)}$) consists of the components either impacting *or* depending on the entity (those which appear in its input or its output). Components out of the entity and its context are the **external world** of the entity, noted $E_{(\not\sim)}$. These

components neither impact, nor depend on the entity, that is, they do not have any relationship with the entity.

(Any input, output, transfer, context and external world of an entity can be considered an entity, too.)

An entity along with its context constitute an **extended entity**.

(*Two entities are independent of each other if and only if neither appears in the context of the other* or, in other words, neither is a subentity of the construction made up of the other entity and its context.)

Considering the behaviors of a system, a change in a behavior with the change limited to the given entity is called an **internal change** of the entity, while its value taken in a given moment is an **internal state** of the entity. Similarly, the change limited to the context is the **external change**, while the state limited to the context is the **external state**. The change and state limited to the entity along its context is called the **extended change** and **extended state** of the entity, respectively.

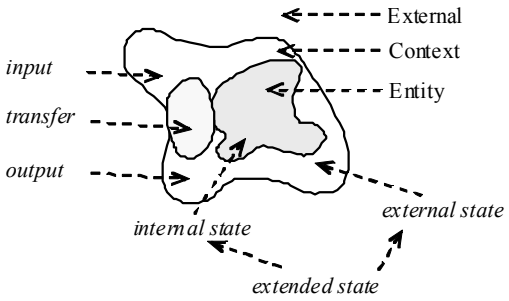


Figure 12: An entity (component) with its context.

(Specifying the extended or internal change of an entity results in the abstract model of the behaviors.)

(*An entity specifies a double abstraction of the behaviors*, where the first abstracts from the external world of the entity, while the second performed on the result of the first abstracts from the context of the entity.)

(***Two subsequent domain abstractions specifies an entity of the system*** in the following way: Impacts between components may dissipate during both abstractions. Consider some components of the composite. The components still in relationship with the selected components after the first

abstraction are together considered to be the extended entity, while the ones still in relationship after performing the second abstraction are together considered to be the entity. Such a method of finding an entity is called **entity outlining**.

V.
Changing in Time

Event, Transition and Modifier

Limiting the change of a system to the start and end point of an interval (i.e., to state combinations taken in a "base" and a "studied" moment) results, according to earlier definitions, a *state change* or *step*. The state change may even be identical, when the states in the base and the studied moment are equal, that is, no *actual* state change takes place. Non-identical state changes are called **events**, attributed by a **moment** or time (the latter or "studied" moment) and, in case of composites, a **location** or **source**, which is the changed entity.

The followings assume employing *deterministic specification* method when investigating events and behavior.

Ignoring the moments of a state change yields in the pair of the start and end state (s, s'). Generally speaking, an element pair ($(s, s') \in S \times S$) of a state set S is a **transition** (written as $s \mapsto s'$), treated a two-moment (quantum) system, where s is called the **initial** or **source state,** and s' is called the **target state**.

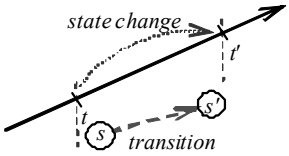


Figure 13: State change and transition

Pairs of non-empty state sets are called **abstract transitions** (and can be described by sets of specific transitions, i.e., by a non-empty subset of $S \times S$).

(State changes can be specified by state transitions with proper positioning, that is, by specifying source locations and moments.)

Mappings of the form $m : S \to S$ are the **modifiers** or **transformations** of the state set *S*.

(*A state transformation specifies an abstract state transition*, since it (uniquely) determines a target state for each source states.)

Transformation: Transition and Action

In the precedings we have already found that abstractly defined systems can be rewritten into the form of transition functions, which tell the changes relatively to a fixed moment. The following statements are therefore true:

A startpoint-based quantum system specified by a transition function can be rewritten into the form of a pure transition function, that is, into a system with a transition function that only requires to know the state of the system in a given moment in order to specify the state taken in the next moment (if such a moment exists). Changes in such a system appear as transitions, while the "pure transition function" appears as a state transformation, i.e., as a mapping $m : S \to S$ specifying, for a state *s* and a given moment, state s' taken in the next moment ("state successor").

(The statement can be extended to any arbitrary quantum system, so every quantum system can be specified by a pure transition function.)

(Startpoint-based quantum systems can be specified by the state taken in the start point (in deterministic case) *and the appropriate state transformation.)*

The change of a given entity of a system is determined by the entity itself and its context (input), while that change may impact its context (output). From the point of view of the entity, the changes of the complete system may be grouped into two components, based on the impacts related to the entity. One of the components is the change of the internal state and is called **transmutation** or internal transition, determined by the entity and its context (or more precisely: by its input). The other component is the impact

of the given component on the context (or more precisely: on the output), called the impact or **action** of the given component.

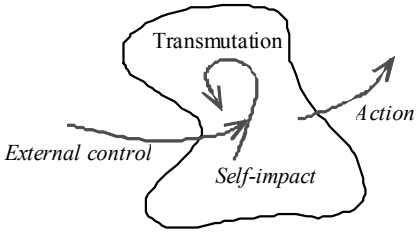


Figure 14: External control, transition and action

Therefore, a state taken in a given moment by a system's given entity may impact a later state of its context (action) and, in accordance with its input, may impact a later internal state of itself (transition).

Interactions

Identifying impacts within a system decomposed into entities let us give a special definition of changes. The specification with a *system of entities* decomposes the complete system into entities, which may have relationships between each other (may impact each other), and where the entities determine their changes based on their input and own internal state, and, accordingly, impact their context based on their input and own internal state.

When employing the approach of entity systems, the change of the complete system can be specified as a given form of *definiteness*, where changes occur "automatically", based on the internal transformation rules.

In case of connected entities, the transmission of certain impacts are considered immediate, such as when an electric bulb immediately starts emitting light upon turning the switch on, or when a bowl of a pair of scales raises upon the other is pressed. There are, however, relationships when a kind of indirectness may pop up such that the impacting and/or the dependent entity has to initiate the transmission of the impact. Therefore *active and passive entities* may be differentiated regarding the given indirectness present in impact transmission. Impact transmission is initiated

by the active entity, while the passive entity only reacts upon the questions regarding its state and "bears" the transformations.

(For example, an "active" applicant "entity" submits an application to an office, thereby initiating the execution of a given process. The applicant may ask for information on the result of its application (where the result is considered the abstract state of the application). In this situation the applicant is the active party. On the other hand, however, the applicant might be notified of the result, making the applicant the passive party in this scenario. In case of a successful application, the applicant gets access to the given resource after contracting, making him passive again.

The actual state transition of an entity (what is an internal event) appears to be an (external) event for external entities that may force them to change their own states. In case of connected entities, a change may, therefore, generate or initiate a series of events and transitions.

In a system of entities the following apparatus is required to *specify transitions*:

- The entity
  - has to answer questions regarding its abstract extended state; and
  - has to be able to query the abstract states of the entities within its context (or at least those constituting its input).
- The entity has to enable modifications to its abstract state. This may be done via
  - direct request for state change; and
  - indirect reaction to certain signaled events.
- The entity has to indicate the change of its abstract state toward its output.

Bearing an impact consists of the following steps for *active entities*:

- The entity queries the (abstract) state of its input.
- The entity may change state depending its input and its own abstract state.

- If an “essential” state transition happened, this may be published (“signaled”) to its output entities.

Enforcing an impact consists of the following steps for *active entities*:

- Either indirectly:
  - The entity publishes its (eventually changed) abstract state in the form of an external event.
  - Then the dependent entity tests its context and performs the necessary modifications.
- Or directly:
  - The entity, based on its own state, and on the state of the dependent entity and its context, performs the modifications of the dependent entity directly.

The two methods of enforcing an impact differ in indirectness (or degree of freedom) and in the strength of intervention. The first method provides a greater level of autonomy and decision scope for the dependent entity, while the other forces channeling (the dependent) into an explicitly defined state bed. Therefore the first method is more variable, yielding in a more diverse system, while the second is able to maintain stricter constraints. (!)

For *passive entities*, the way of *enforcing the impact of an entity* is as follows:

- The entity responds to queries regarding its abstract state.

For *passive entities*, the possibilities to *bear an impact* affecting the entity are the followings:

- Indirectly: The entity may receive (external) events, and in response to those events:
  - It may change its state according to its own state and that of its context.
  - It may trigger additional impacts on its output by actions (either indirectly through an event or directly by a modification operation).

- Directly:
  - The entity enables modifying its state externally.
  - It may directly or indirectly trigger additional impacts as consequences of the external modification.

Therefore the system of entities can be specified as a definiteness that is basically a system of state transformations triggered by given conditions. Indirectness, the greater degree of freedom employs more (event) notifications, instead of forcing the state change of other entities. Higher degrees of freedom allow for higher diversity.

(***Specifying entity interactions.*** A system decomposed to entities, that is, a system of connected entities, can be defined with the following apparatus consisting of three tools. An entity in the system of entities:

- (event:) in response to internal or external events
- (internal transformation:) may change state depending on its own state and that of its context,
- (action:) and it can notify its context about that change, or may enforce the modification of its context.

)

The behavior of connected entities, i.e., the **interaction** (or **collaboration**, in other words) among the entities (components) of a system is defined as (1) information propagation from some given components (called the impacting components) toward some components (called the dependent components); and (2) the state transformations performed on dependent components, in the course of which impacting components publish their abstract extended states and, in a given (abstract) way, may perform interventions on the states and changes of dependent components. Interactions represent the impacts among the components, so the terms interaction and impact are considered equivalent.

(Interaction, *from the point of view of impacting components*, is the *publication of their abstract extended states* (the component does not "know" any more information). The *abstract state* of a component of a given nature can be, therefore, interpreted as an *abstract impact*, in which we do not consider the dependent component its state transitions triggered.)

(Interaction, viewed from the *aspect of dependent components*, is *the bearing of the impact*, which may cause changes in the states of the components. The change in a component may extend only to itself and its output. Given a component, *bearing an impact of a given nature* is an *abstract impact*, where we can ignore the impacting component and its state.)

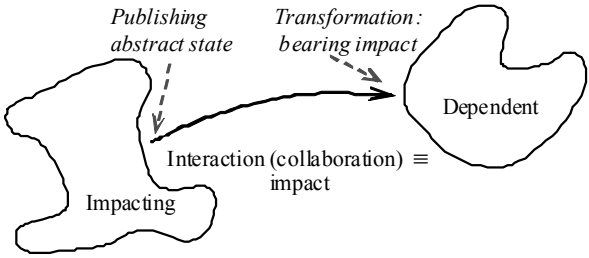


Figure 15: Interaction (impact).

Activity

Change in startpoint-based quantum systems can be specified as a series of transformation steps from the state of the initial moment to the next moment and, similarly, from the state of any moment to the state of the subsequent moment. Transformation is considered a zero-duration operation that, based on the given (extended) state, determines the state valid in the studied moment.

The composite transformation (function) of a transition can be divided into a series of *atomic transformation* steps (i.e., a composite function made of functions considered to be atomic). During performing a composite transformation (such as at an intermediary step of a series of transformations), the entity or the joint system of entities may only be in a partially changed, that is, in an inconsistent state. The state of the entity is undefined while performing the transformation and, therefore, may not be used to determine the necessary transformations. For this reason, the entity or the entities may not receive events or accept modifiers while composite transformations are being performed.

A transformation can be converted to an operation having a non-zero duration if, for example, the system “waits for” the next or a later moment of the quantum system to come after the transformation. The pair of a zero-duration and therefore uninterruptable transformation and the waiting for the subsequent cycle is called **activity**. Transformations may be composite, that is, made of the dependent or independent composition of its subtransformations. Transformations may be empty (identical): the "most atomic activity" is the waiting for the next cycle.

Composite activities may be constructed as a series of atomic activities. A composite activity can be interrupted after the cycle of an atomic activity, for example, due to the impact of an external event.

The strongly monotonic change of an entity (a change of a cumulative or regressive value of it) depending exclusively on time, and the corresponding value limits, are together called **timeout**, if passing the value limits may yield in impacts on the entity or its context. An **atomic timeout** is a single value limit.

(Composite timeouts (those containing multiple value limits), however, can be defined as a series of atomic timeouts. *An atomic timeout appears as a waiting for a given moment*, but indeed it is the image of a significant change (abstract value change) of the earlier mentioned cumulative or regressive quantity.)

(In quantum systems, the smallest timeout is the waiting for the next moment, as the system is not defined in “earlier” moments. *Passing a timeout corresponds to an event*, where the actual (abstract) state transition from the base state is, the act of passing the value limit.)

(The beginning of an activity specifies an event. The end of an activity, however, specifies an event, too. Activities are related to time periods (durations), since they contain waitings, while transformations are “immediate” operations.)

(*Activities specify state pairs*: upon starting an activity, the executive entity steps into an abstract state corresponding to the execution of the activity, and leaves that state when exiting the activity, and therefore it steps into another abstract state. *The terms activity and abstract state, therefore, correspond to each other*.)

The relation between activities and abstract states implies that the change of certain entities may depend on whether the given entity is actually performing an activity, or an activity has been completed or interrupted. Waiting for the completion or interruption (together called: end) of activities is called **synchronization**.

Abstract State, Event and Modifier

Let's perform the following on a given component of a system—for example, on the state of an entity extended with its context. Let's group actually taken states into a finite number of disjoint subsets, called classes. Should we result in two or more classes, perform another classification in the next step by aggregating certain classes. The grouping can be performed in a finite number of steps, since we have allowed only a finite number of classes in the first step, and the number of classes has to decrease in the subsequent steps. The system of classes and subclasses grouping the states of the partial system (that we've started to classify) is called the **stepwise finite partitioning of the state of the partial system**. This is a simpler version of the stepwise approximation by finite alternatives for determining abstract states.

(The *stepwise finite partitioning of states is an aspect*, if states are considered state combinations taken in given moments. For the aspect, *i0* specifies specific states, while the last $i\infty$ step results in the complete set of actually taken states.)

(*A single abstract state other than the one specifying the complete set of actual states determines a stepwise finite partitioning*, where the group corresponding to the abstract state s appears as a class in the only intermediary level, and the group $\neg s$ of the difference of the complete set and the states belonging to the given abstract state appears as another class.)

Since state partitioning can be performed along various aspects, complex relations may feature the abstract states, as noted below.

(Certain abstract states of the entity may be grouping or collective terms ("super states") of some other (less abstract, i.e., more accurate) abstract states. Therefore, for a given component of the system, multiple abstract states being consequences of each other may be valid at once.)

(Since multiple aspects can be defined, multiple abstract states (being not in condition or consequence relationship with each other) may be valid for the given component of the system. These abstract states are the results of different partitionings.)

(All classes belonging to an abstract state, which were yielded in the preceding step of the partitioning are mutually exclusive (disjoint), and this way the fulfillment of the (grouping) abstract state implies that the fulfillment of an abstract substate of the partitioning causes the unfulfillment of the other classes. This exclusion principle also stands directly for the classification yielded in the very first step of the stepwise finite partitioning.)

Abstract states are closely coupled with Boolean functions, as those specify selections.

(***Each abstract state can be mutually mapped to a given Boolean function operating on the set of actual states***, where the function assigned to a specific abstract state returns true for all states belonging to the group of that abstract state, and otherwise returns false.)

(A Boolean function defined on the actual states is appropriate for defining an abstract state. All Boolean functions defined on the actual states specify a simple aspect.)

(*Logical operations can be used to produce additional abstract states from some initial abstract states.*)

Several different systems of abstract states can be specified using the set of actually taken states of an extended entity, however only those are "essential", which can affect the behavior of the entity and its connected entities. From the aspect of the entity's context, the "essential" abstract states are those, which impact the entities of the context. These are called **public abstract states**.

The entity, however, can perform a certain series of internal operations, or even perform multiple series of operations simultaneously. Specifying the series of internal operations may also require the identification of certain abstract states—for example, accessing abstract states indicating the end of two activities (synchronization) may be a condition of a given state transition. These **internal abstract states** only indirectly appear as external states, but are appropriate, nevertheless, to describe the internal change of the entity.

Studying from its context, the change of a given entity appears as an *external event*, while its own change is seen as an internal event from the side of the entity. From the aspect of the context, therefore, it is eligible to specify only the "essential" abstract state changes of the entity as external events, that is, those that may cause change in the context. These are called **public events**. **Internal events** do not necessarily trigger public events, but are eligible for describing the internal processes of the entity or, in other words, for giving an abstract, conceptual level specification of the entity's change.

Entities can be connected directly in the system of entities: the entity actually performing a state transition performs the necessary modifications on its dependent entities directly. The indirect method allowing for greater variability is basically a system of "notifications": the changed entity publishes the ("essential") changes of its state in the form of events, and the entities "listening to" the notifications may perform the necessary state modifications in response to the events. The indirect method has to enable the coupling of or "weaving" the impacting and dependent entities.

In case of a system specified by a pure transition function, the *entity* can be described *from the aspect of its context* as follows:

- The set of public abstract states, which may also impact entities in the context passively (abstract states can be queried).
- The set of public abstract events—for example, published abstract state transitions (those "told" other elements)—which may impact the context actively.
- The set of externally visible abstract states. The impacting entity of the context can directly trigger state transition in the entity with the help of a modifier.
- The set of "listener" abstract state transformations. With the help of listeners, the entity can bear the impacts usually appearing in the form of events.

The *internal behavior of the entity* requires the specification of some additional elements:

- Internal abstract states
- Internal abstract state transformations
- Activities
- Internal abstract events (e.g., indicating internal state transitions, timeouts and the end of activities)

So connecting entities indirectly can be done by specifying abstract transformations triggered by abstract events.

Control: Operation in Algorithmic Environment

State changes and object interactions of real world objects happen *automatically*, governed by the laws of the real world. Should air temperature decrease below the freezing point, the water of the lake freezes; or stormy wind may broke branches of trees and so on.

When representing objects in algorithmic environments (by a Turing-like machine, such as a computer), automatism of the laws has to be simulated as required by the algorithmic environment. Algorithms are common in that all those specify a method of achieving a given goal, therefore algorithms are based on certain orientation or *deliberateness*. So the automatism of laws has to be simulated by deliberateness in algorithmic environments. In the example above, air has to publish its temperature to the lake that will perform a transition into "frozen" state when temperature has decreased below a given value. An alternative to this is when air instructs the lake to freeze upon air temperature decreases below that certain value. Orientation can be reversed, however, and the lake can query the air temperature and may “decide” on freezing itself. The first and third versions yield in a more realistic model, while the second is more effective, since it’s closer to the nature of algorithmic environments.

The apparatus for implementing deliberateness is one of the most fundamental principles of cybernetics, named *control*, that yields in impacts affecting the controlled thing. Control is an asymmetric relationship: the controller’s *control* over the controlled thing.

A given impact can be represented in two ways in algorithmic environments: either the controller impacts the controlled object, or the

object queries the (abstract) state of the object impacting it, and then performs the necessary state transition.

The followings study interactions, which may be implemented using the traditional control structures of function calls and returns. In this case, the caller is the controller that passes control to the called subprogram, which is the controlled object. Both caller and called components are considered operating, since the caller does not interrupt, but only suspends its operation.

In case of the call-return control structure, considering the given operation (subprogram, i.e., function or procedure) of the called object, the following possible interactions exist between the controller and the controlled object:

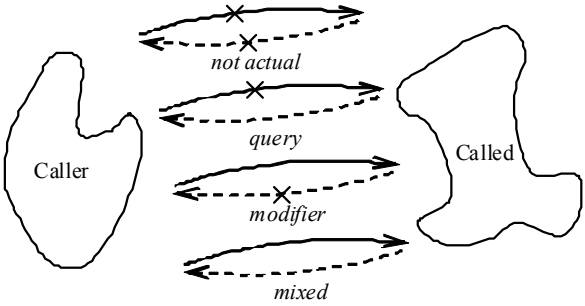


Figure 16: Possible types of call-return interactions.

- The operation does not return any value, nor does it cause side effects (where waiting is considered side effect, too). This is indeed an empty (identical) transformation that may not be considered actual interaction.
- The operation does not cause side effects, but returns a value. The object is only aware of its own and its context's (input's) state, therefore this type of interaction means the *querying*, *getting* or *retrieving* of the abstract extended state. This is unidirectional information propagation from the called to the caller.
- The operation does not return any value, but causes side effect. Such operations are called *transformations*. *modifiers* or *setters*.

This type of interaction is unidirectional information propagation from the caller to the called.

- The operation causes side effect, while also returns a value. This type of interaction can be interpreted as the combination of a modifier and a query, and manifests bidirectional information propagation (information exchange). Since returning a value means determining a single, non-repetitive (non-iterative) value, this *combined* kind of interaction can be specified by the ternary of an initial modifier, a query and a final modifier element, where the query determines the value to return, and either the initial or the final modifier may be omitted.

VI.

Internal System

Internal System, Structure

System, in our principle, denotes the process, the completeness of change; there is no additional external world outside of the system, or any "external system" with which our system would be connected. So far our studies highlighted a single entity of the system and analyzed the change of the system within the course of the highlighted entity. This method can be extended such that the change of the complete system is specified as the definiteness of collaborating entities impacting each other. In order to investigate the *internal system*, we have to define in what *relationship* the entities are with each other.

Impacts within the system may be modified as the result of an abstraction. Sometimes phantom impacts (superstitions) may also appear. If an abstraction does not extend the domains of the changes of the elements or, in other words, *domain abstraction* is applied, impacts may at most weaken, or sometimes disappear, that is, dependencies may decrease. Domain abstractions are suitable to "outline" an entity belonging to a given location of the system.

Domain abstractions have therefore a special role from the aspect of relationships between components. Apart from $i\infty$, aspects containing domain abstractions are called **domain aspects** or **structuring aspects**.

Impacts are transitive, so connected components can be grouped into disjoint "islands" where non-connected, independent components reside on different "islands". Domain abstractions decrease dependencies, enabling us to create additional closed groups within the islands. Elements of these groups are, in this abstraction level, independent of the elements of other groups. Additional domain abstractions, in turn, let us defining additional

subgroups and so on. Both islands, both the groups and subgroups of the islands appear as entities of the system.

(***A structuring aspect generates a hierarchical decomposition of the system***; non-connected groups of the same level are disjoint, and components may be decomposed into additional independent subcomponents at a higher value of the aspect.)

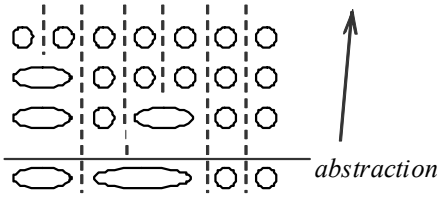


Figure 17: A system gets decomposed into independent entities by applying a structuring aspect

Consider the disjoint sets produced by a domain abstraction as decompositions. Projecting those back to the original system yields in a containing hierarchy, called the **decomposition of the system by domain abstraction**. The system of hierarchical entities and subentities yielded as the result of a decomposition by a structuring aspect are called the **aspect-aware structure** of the system.

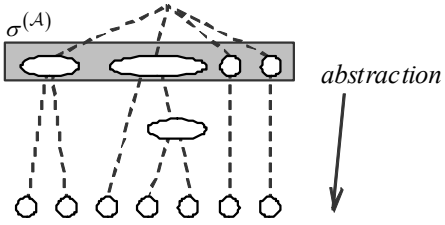


Figure 18: Structure as hierarchical decomposition

A **system is** said to be **structurable in an aspect** when dependencies decrease upon applying a structuring aspect, that is, the aspect at an intermediary abstraction level decomposes a component of a preceding level into multiple subcomponents.

Typing, Conceptual System

Following the abstractions of a structuring aspect, the system gets split up to independent components such that those components can be further decomposed into its independent subcomponents. At the same time, more and more similar, but initially different details of the original system will be sorted into the same group by higher and higher abstractions of an (arbitrary and not necessarily structuring) aspect. The number of the groups decreases, while including greater and greater details of the system. The ever-comprehensive types form a hierarchical system, too, corresponding to the reverse order of applying the series of abstractions. The bottom of the hierarchy contains the components of the initial system.

The idea of concepts and the technique of positioning makes it possible to define the identical nature of the different components of a system: Two (comparable) systems are treated **conceptually identical** if there is a positioning of one of the systems that yields in a system equatable to the other (notation: $\sigma_A \widetilde{\Leftrightarrow} \sigma_B$). Should a positioning of a concept σ_A exist for which σ_B is a condition, then σ_B is said to be the **conceptual consequence** of σ_A; noted $\sigma_A \widetilde{\Rightarrow} \sigma_B$. Concept σ_B is a **more abstract concept** than concept σ_A if the first is a conceptual consequence of the second, but they are not conceptually identical.

Conceptual identity helps extending the definitions of difference and accuracy to concepts: The aspect-aware **conceptual difference** of the concepts of a given system is the smallest of those values of the aspect where the concepts become conceptually identical upon applying the abstraction related to the mentioned values. **Conceptual accuracy** or, in other words, **analogy** of concepts, interpreted also in a given aspect (that is, the aspect-aware conceptual accuracy or analogy), is the abstract image resulting from performing the previous abstraction.

(The conceptual accuracy or conceptual analogy of the concepts of a system is a concept of that system. The conceptual accuracy of the concepts of a system is the conceptual consequence of the given concepts.)

Given the concepts σ_A, σ_B and σ_C of a system (for example, components of a composition), σ_A and σ_B is said to be **more analogous**

concepts in one aspect than σ_C, if the analogy of σ_A and σ_C, and that of σ_B and σ_C both are more abstract than the analogy of σ_A and σ_B.

Should we treat the components of a system as concepts, the more and more abstract images of those concepts become more and more analogous to each other, and more and more times will a positioning of an abstract image correspond to another abstract image. An aspect, therefore, defines an abstraction hierarchy, where components becoming conceptually identical at a given abstraction level can be classified into the same group at that level.

Considering the disjoint components of a system in a given aspect, grouping the images of components belonging to one or the other abstraction level such that the grouping is driven by conceptual identity is called (**conceptual**) **typing**. (The result of typing is a hierarchical grouping.)

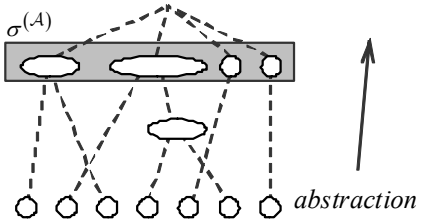


Figure 19: (Conceptual) typing.

Typing, similarly to structuring, specifies a hierarchical grouping of the components. Structuring decomposes a system into independent components, while in case of typing, however, higher-level abstractions define ever-greater groups and therefore more and more abstract concepts. The ever-stronger abstractions in an aspect simultaneously decompose and group the system.

Consider those systems resulting from performing abstractions in some given aspects on comparable systems $\sigma_{i\in I}$ (i.e., $\Sigma^{(\sigma_{i\in I})^*}$), which belong to the same equivalence class in terms of conceptual identity. The set of resulting concepts is the **conceptual system** of systems $\Sigma^{(\sigma_{i\in I})^*}$ in the given aspects (also called aspect-aware conceptual system).

(Since concepts of a system are comparable, concepts of a comparable system are comparable, too, and therefore even the concepts of a conceptual system are comparable. Elements of a conceptual system are not necessarily concepts of every system, since concepts are produced (by positioning) from actual concepts of a system.)

As concepts and as the positionings or, that is, applications of concepts also produce comparable systems, those may also serve as the basis for constructing macrosystems (via conjunction or disjunction, for example). This leads to say that certain components—themselves being concepts, too—of a system can be described by some more atomic concepts.

Impact Compensation, Cohesion

By applying a structuring aspect, certain components may disjoin from each other sooner than others may. Aspects are therefore suitable to specify the connection strength or cohesive force of components.

The **cohesive force** between specific components of a system in a structuring aspect is that smallest value of the aspect at which those components become independent, once the abstraction belonging to that value has been performed. (The cohesive force between (or of) independent components is minimal, that is, equal to $i0$). Given a structuring aspect, components of stronger cohesion disjoin later.)

The definition enables us to differentiate between components *more closely connected* and *more loosely connected* in a given aspect within the system.

Consider an entity impacted by another. The difference between the initial state of the dependent entity and the state of it yielding from bearing the impact may disappear after performing an appropriate abstraction, causing the two states to become identical. That is, abstractions may absorb or "compensate for" impacts affecting the entity. In case of another impact, a stronger or weaker abstraction may be necessary to "compensate for" the impact, meaning that the strength of the abstraction to be performed is suitable for measuring the impact borne by the entity.

The strength of an impact in a given aspect affecting an entity (called the **impact strength**) is that smallest value of that aspect at which the entity and

the version of the entity resulting from bearing the impact becomes equatable by performing the abstraction belonging to that value. Accordingly, impacts being stronger and weaker in an aspect can be defined. (Impact strength is the difference of the original behaviors and those resulting from bearing the impact, both considered as comparable systems.)

(The ***impact strength in an extended real aspect is distance*** interpreted on the behaviors belonging to various parameters of a selected entity of an abstractly specified system.)

Impact strength is therefore understood as the difference of the original and the modified change, so that the higher the difference, the greater the impact.

State transitions of an entity of a system or, in other words, events of an entity of a system may impact the context of that entity. So to say, the behavior of the system may differ from the system's behavior belonging to the constant change of the entity. **Relevance of an event** (**essentiality of events** or **significance of events**, in other words) in a given aspect is that smallest value of that aspect at which the related abstraction makes the two mentioned behaviors equatable. This allows to define the aspect-aware **relevance of an external event** and to differentiate between more relevant and less relevant events. From the point of view of an entity, an external event is more relevant in a given aspect if the event causes, in that aspect, greater difference in the change of the entity.

Thus what the relevancy of an event or external event is correlated to is not the frequency or infrequency of the event, but the extent or measure of the impact caused.

As the above definitions imply, concepts of impact strength and cohesion share a common approach. This can be imagined as certain impacts of those affecting an entity may terminate a relationship between some components of the entity, and that the force (or impact strength) necessary to terminate the relationship of components more closely connected in an aspect is greater.

Consider, for instance, a room with two chairs placed next to each other so that one touches the other. By pushing one of the chairs toward the proper direction (impact), the chairs may move together. All other components of the room are independent, while there is some cohesive force between the

two chairs. When pushing the studied chair into the opposite direction, its movement, however, will be independent of the other chair. Here the impact had the mentioned cohesive force dissipated. This impact might be compensated by replacing the image "there are two chairs next to each other in a room" with the more abstract image "there are two chairs in a room". The chair might break under the pressure of some heavy force. The break is likely to be aligned along joints, possibly causing the post or the arm of the chair to be knocked off. The impact could be compensated only with an even greater abstraction, such as "there are the components of two chairs in the room". Components of the chair-post are more closely connected with each other than to other components of the chair, meaning that the cohesive force is greater between them, and therefore much stronger force is required to separate those components. This series of activities (i.e. smashing the chairs), resembling a decomposition, specifies the structure of the system in an aspect (or, in other words, specifies an aspect-aware structure) describing the chairs as *chairs(chair (arm, back, seat, post1, post2, post3, post4), ...)*.

Implicit and Explicit Relationship

Relationship between two entities of a system is the impact of one of the entities on the other and/or the impact of the other on the first. Impact appears as a conditional ("if-then") change in the behavior (or change) of the dependent entity: *if* the impacting entity shows a specific behavior, *then* it channels the behavior of the dependent entity into a specific bed.

For the external viewer, impact appears as a certain joint change, that is, the change of the impacting entity implies change in the dependent.

If an entity conceptually identical to a given entity of a composite system appears in another location in the system, then both entities may react identically to the impact of their common or shared context. What the external viewer recognizes is that entities change synchronously, in parallel. The affected entities seem to maintain a relationship with each other, but indeed, however, there is not necessarily any actual relationship between those. The impacting "external entity" also recognizes some kind of relationship between the two entities, since both react identically to the impacts caused. The same stands for the case in which two entities identically impact a third entity considered "external".

(***Relationship as a consequence of a system*:** Given relationships between given entities specify a consequence of the containing system, defining the states of the entities up to the extent of the accuracy of impacts and the result-constrainers of the impacts, and, furthermore, ignores those components of the system which are not connected to the mentioned entities. This means that ***impact is information regarding the system***, since the constrainer of an impact may not be the arbitrarity.)

(***Analogous entities connect to other entities in the same way.*** Entities being (conceptually) analogous connect to each other and to "external" entities in the same manner, if applying the aspect of the conceptual analogy does not modify the impact.)

(The previous statement reversed: ***Entities connected to a given entity in the same manner*** (i.e., all of them being in impacting and/or in dependent role) ***form conceptual analogy***, where the aspect of the analogy is determined by the abstraction the relationships specify.)

Conceptually analogue entities may connect to other entities in the same manner, and may produce impacts of the same nature or may react to impacts in the same manner, yielding that external viewers sense those as being connected. Such relationships are called **implicit relationships**. To distinguish implicit relationships from actual relationships between entities, the latter are also called **explicit relationships**.

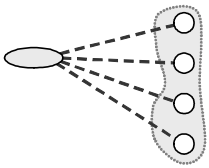


Figure 20: Implicit relationship as conceptual analogy.

Let's replace an entity of a system with another. The difference is seamless (or in other words, cannot be realized) for conceptually identical entities. In case of conceptually different entities, however, the behavioral change can be compensated for with a suitable abstraction, which has to dissipate the change of the relationships of the entities, including state

transitions of any other nature and the impacts of any other nature caused by those transitions..

(***When replacing an entity of a composite with a more analogous entity, the change of the system will differ less from the original change*** in the aspect of conceptual identity. At this point we can declare the ***principle of substitutability***: considering a given abstraction, when substituting an entity with another, the abstract images of the behaviors remain equatable if the abstractions of the two entities are conceptually identical or, in other words, the conceptual analogy of the two entities correspond to the level of abstraction.)

Collaboration

Certain events of an entity may impact another entity, and those impacts may be propagated indirectly to more and more entities. The impacting entity realizes this phenomenon as the entities were to react on the event collectively. The external entity senses the connected, related entities together, as a single unit or *macroentity*.

Collective reaction implies orchestrated behavior (or orchestration, for short), however orchestration is actually the set of relationships within the macroentity. Since internal relationships do exist in such a macroentity, some internal cohesion also do exist and vice versa, the stability of the macroentity, that is, its capability to resist external impacts, is determined by the strengths of internal relationships.

A macroentity with internal relationships is, of course, more than just a simple union of its components, and this extra is actually provided by the connection types belonging to the relationships, and these extras enable the macroentity to present its context organized or orchestrated actions and reactions. Collaborating entities may share high external loads (impacts) and, in addition, may generate greater impacts collectively.

The entities working in orchestration due to their internal relationships are collectively called **organization**, while the orchestrated operation is called **collaboration**. Orchestrated behavior means that entities, based on their internal system of relationships, collectively impact the external world directly or indirectly, and react on certain external impacts collectively, too.

(An organization is a macroentity with an appropriate context and external world. Collective actions and reactions have the organization appearing as such a single and uniform entity for its context that impacts the entities of the context in a given manner and reacts on the impacts of the context's entities in another given manner.)

(Organizations may not incorporate entities independent of any other entities within that organization, therefore *the components of an organization are connected to each other*.)

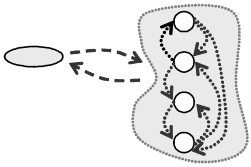


Figure 21: Organization and collaboration.

The **closure of macroentities** or **closure of concepts** is an abstract image of some given macroentities or concepts such that the image sorts the entities and concepts connected to the context in the same manner into the same group, and assigns those a single abstract image.

(*By introducing closure, we can ignore the exact internal structure of the enclosed entity*, considering it as a black box. *Closure is the abstract model of the entity* since it can be specified using the appropriate functions. *The closure is a constrainer of the entity and, for unindependent entities, an information regarding the entity.*)

(*Substitutability and closure:* Should an entity be replaced or substituted with an entity the closure of which is equal to the closure of the original entity, then the behavior of the components other than the two entities (i.e., the context and the external world of the two entities) remain the same.)

The organization may perform complex series of internal actions in reaction to external or internal events. The activities are implemented by the activities of the internal entities and the complex system of relationship between those entities. Relationships ensure orchestration or, in other word, organizedness, causing the activities and actions of the entities and the

reactions of the entities to external events to appear uniformly. Upon substituting certain components with an appropriate, conceptually analogue entity, the internal behavior of the organization remains the same in spite of the substitution, yielding in that the organization shows the same picture to its context (that is, the closures of the two systems are identical).

The enclosure of an entity within a macroentity is called a ***role***. (Role is the abstract model of the entity. In addition, role is a constrainer of the entity, as well as, for unindependent entities, information regarding the entity.)

The same role can be taken by more than one entities within a macroentity.)

(***Substitutability regarding roles:*** Substitution of an entity with another one of the same role within a macroentity does not modify the closure of the macroentity.)

(It's worth to make a note about organizations, stability and degree of freedom. An organization can uniformly act on its context and uniformly react on the impacts of its context. The organization requires maintaining a given structure for this purpose that, considering participating elements, means taking given roles. Roles are, however, constrainers at the same time, and therefore decrease the variability or degree of freedom of the entity taking the role, as well as those of the organization. So the stability of an organization *may* yield in the inflexibility or bound structure of both the organization and its participants (members), making the organization unable to react properly on a new impact. The mentioned inflexibility can be resolved by easily permeable roles, assuming some internal dynamism. This makes inflexibility or degree of freedom (and even external stability) measurable (and open new ways toward organization modeling?), and this is what may appear for the external world as stability and dynamism at the same time.)

Entities in the same role within a macroentity are called **collection**.

(Collections are macroentities. Items of a collection are substitutable with each other without affecting the internal behavior or the closure of the macroentity.)

The duality of explicit and implicit relationships is reflected in collections. Since collection items take the same role, some conceptual analogy must be also present. So, depending on the point of view, either the items take a role collectively, or the roles are multiplied.

Now create an abstract image of a macroentity, in which entities (produced by a given decomposition) are substituted with their closures. The resulting abstract image is the **abstract structure of the macroentity**.

(A makroentitás absztrakt struktúrája a makroentitás absztrakt modellje.)

(***Macroentities can be defined as relations:*** The abstract structure of a macroentity determines the behavior of the macroentity down to the level (or up to the extent) of roles. The macroentity can be described with an n-ary $\Re(\rho_i : E_{i_j})$ where n-ary $\Re$ is called the closure of the macroentity, ρ_i are the roles of the macroentity, and $\rho_i : E_{i_j}$ specifies the entity taking the given role.)

Context

Changes in an impacting entity may imply the change of the dependent entity. Impacts have to get to the dependent entity somehow (in active or passive manner). Impact may propagate directly or indirectly, via the transitivity of a transmitter or some transmitter items.

(Connected entities collectively define a macroentity.)

(Considering the macroentity of connected entities, the closure of an entity regarding the connection is the role of that participating entity.)

(Considering connected entities, relationship appear as a constrainer for dependent entities, and as information regarding the entities.)

(Any given relationship between entities can be described using an n-ary such that its name refers to the relationship and its roles are the roles of the entities.) *A relationship of a given nature can be represented by a relation where the entities currently taking the roles described by the n-ary are ignored.*)

For directly connected entities, each entity accesses its connected entities, and therefore may perform transformations on those and, in

addition, may query the abstract states of other entities, so to perform its own state transitions based on the impact of those entities.

Transmitters are less significant in case of *indirectly connected entities* than the impacting and receiving parties. When the role of the transmitter entities can be completely ignored, this relationship is much like a direct connection between the essential entities. For example, consider a light source illuminating a wall. In this case the individual "photon entities" are generally irrelevant, and it is sufficient to take the collective behavior of those into account.

In case of indirect connections, however, transmitters often have the possibility to modify the impact. That is, in this case a transmitter *medium* needs to be defined, which is responsible for transmitting impacts and, on the other hand, may intervene into the impact mechanism.

The followings introduce a simple yet powerful method, based on modeling the "context" as a macroentity, to describe impacts of either directly or indirectly connected entities.

The connected entities actually reside in the context of each other. Each relationship is a macroentity and a set of relationships defines a more extensive macroentity.

A macroentity that may contain entities satisfying some given constraints is called **context**.

Context is not just the union of entities, but in itself it is also of the nature of entities, and therefore may have given properties and states, may react on certain events, may have transformations and activities, and may generate events.

Context, in addition to the above:

- contains entities located in the "given context";
- is responsible for transmitting indirect impacts, that is, it is of the nature of *transmitter media*; and finally
- may modify individual impacts and may intervene into impact transmission.

Imagine a game played in a labyrinth consisting of rooms between which players can move around. Rooms may contain various tools. If, for

example, there is a bomb in a room, the player may get injured. A classic question of programming techniques or, more specifically, that of object-oriented technology, is how to model and implement the explosion of the bomb. One of the possibilities is that the explosion of the bomb starts the "bear injury" operation of the player. The alternative approach is when the player is continuously querying the state of the bomb, and in case of detecting explosion, the player modifies itself as necessary to bear the injury caused by the explosion. The problem with these solutions is that the bomb and the player are too closely connected. Such a close connection may pose exceptional difficulties when modifying or extending the game.

Flexibility may be ensured in a third solution partially extending beyond the object-oriented approach and building upon the principle of contexts. According to this principle, the context of the bomb and the player has to be specified, what is actually the room in our example. The bomb will "report" the event of explosion to the room. The context, the room in our example, is responsible for transmitting impacts, therefore it enumerates its items and tells them that "high pressure and thermal shock occurred", also telling the place and time of the event. Context may, for example, take the distance between two entities into account when transmitting impacts, or a "magic room" may decrease or reverse the impact. As part of impact transmission, the context notifies the dependent elements on the impact, which may further modify it—for instance, an armor may weaken the injury.

Consider another example. Assume there is a "straight line" object given in a plane. The object-oriented paradigm says that determining if a point fits the line should be usually done at the straight line. Also, this is the place for the construction of another query to find out if the line is crossed by another straight line object. But where to define the operation testing, for example, whether the straight line crosses an arc (or to test if the line is a tangent for the arc)? The possibilities are the line and the arc. But which one to choose? The answer is again the need to specify the context, that is the plane itself in our case. This is where the rules determining the relative position of plane figures should be defined.

Context, as seen, basically couples other entities, between which some impacts are usually but not necessarily defined. So generally the context is part of the collective context of these entities.

(***Organizations are contexts***, meaning that the concept of contexts is not as strict as the concept of organizations. For example, in case of entities directly not connected, context may transmit impacts only unidirectionally.)

(All macroentities are contexts, just as the complete system is, too. All entities may be considered as contexts.)

This leads to say that context, in itself, is only a synonym of macroentity. The relevant difference between contexts is the difference in the rules (i.e., constraints) specified for their contained elements.

Context, therefore, may be considered as a collection of given rules regarding the entities and impact transmission.

A context may be a constituent of another context. In the previous example, rooms collectively mean the labyrinth as context. Hierarchical contexts are actually compositional hierarchies or, in other words, (macro)entities within a macroentity, yielding that the following properties hold for those.

(***Principle of propagation:*** The properties and rules of the external context (macroentity) are valid in the subcontext by default, however the subcontext may redefine those. The external context "propagates" or offers the entity properties and rules regarding itself. For example, a company (as the context) "may offer" its name, postal, e-mail and web address as the default for its sites, however the sites may define different values.)

(***Principle of delegation:*** The context may "delegate" or transmit received impacts to some of its components (entities or subcontexts) and, moreover, may perform or implement a complex activity by making use of the activities of the components and the impacts between the components. The above-mentioned company, for instance, may transmit the activities of completing a project to its components and entities.)

Decomposition to contexts and subcontexts does not necessarily follows physical locations. Context is the collective space or the "environment" of the connected entities. Subscribers of a newspaper, for example, do not necessarily live in the same town. The example also discovers that contexts belonging to different aspects may be defined by different hierarchies.

Contexts define the *scenes* of changes or dynamics, allowing an entity to enter and later exit the context, much like a player enters a room and then moves along to another.

Structuring Levels

When describing the complete composite system, it can be broken down to related components (i.e., it can be structured), enabling individual specification of the components and the collective rules regarding those. Breakdown is achieved by domain abstractions, yielding in the decompositions of the system.

However, multiple levels of structuring can be identified.

Disjoint sets of components produced by a single domain abstraction are called **regular decomposition** or **regular structure**.

A regular structure defines a system as the adjacency of its components.

The components and subcomponents produced by the aspect containing the domain abstractions are embedded into each other and are called **hierarchical decomposition** or **hierarchical structure** or, in other words, **context-free decomposition** or **context-free structure**.

A context-free structure describes the system as a hierarchy of components and subcomponents, and it can be represented with a tree, i.e., with such a directed graph that has only one startpoint and in which each node can be accessed along only one path from the startpoint.

The aspect containing the domain abstractions produces sets of components and subcomponents embedded into each other. Now assign each index value of the aspect a *method* (even the same for all levels). Items of the same level for which the method assigns conceptually identical systems are considered the same item. The resulting structure is named the **hierarchical typed structure** or **context-sensitive structure** (**hierarchical typed structure** or **context-sensitive decomposition** in other words).

By assigning every level the appropriate identical method, conceptually identical components are considered identical elements at every given structuring level.

A context-sensitive structure can be represented by a directed graph, such that each edge corresponds to an identical element, that is, to elements of the same type. Context-sensitive structures, therefore, can be represented by directed graphs with a single startpoint, in which all other nodes can be accessed along multiple paths from the startpoint.

The last step in a context-sensitive structuring, if all abstractions of the aspect have been performed, produces arbitrary abstract systems from the entities. That is, the result is a mesh, a directed acyclic graph having exactly one startpoint and exactly one endpoint (or leaf element).

Let's consider a context-sensitive decomposition. Looking "downward" toward the direction of the decomposition from the point of view of an element in an intermediary level of the decomposition, a submesh can be seen. If the types cannot be identified, the submesh appears to be a tree and therefore corresponds to a context-free decomposition. Looking "upward" toward the direction of the macrosystems, a certain series of embedding, more and more extended scopes of macrosystems appear.

The hierarchical typed structures resulting from multiple structurings are collectively called **general structure** or **general decomposition**.

General decomposition, as implied, enables performing decompositions along multiple aspects (containing domain abstractions). When studying a given entity or entity type of a decomposition level, looking "downward" toward the decomposition reveals something resembling to a tree that, in case of identities could have been observed, turns to be a mesh or submesh. Since the image of the entity may appear in decompositions belonging to different aspects, looking "upward" toward the macrosystems, a mesh will appear that, by ignoring identities, turns to be a tree.

(***General decompositions can be specified by directed acyclic graphs (DAG).***)

The general nature of the DAG belonging to the general structure means that representing the frames of a system does not requires general graphs, but rather the benefits of the directed and acyclic nature can be leveraged.

Consider the point of view of an entity located on a given level of a DAG. What can be seen when looking "downward" toward its components and "upward" to the macroentities is a context-sensitive hierarchy or, if analogies cannot be discovered by the entity, a context-free hierarchy. In the "upward" direction this means the macroentities along more and more extended, various aspects. Often the only question is whether an entity is a component of a macroentity or not, and whether an entity belongs to the type of the macroentity or not. To address such questions you don't even need to

know the entire hierarchy—it's enough to consider the set of entities appearing in the "upward" direction as a series or a tree.

(The different structuring levels are more and more extended concepts, that is, a regular decomposition is also a context-free, context-sensitive and general decomposition and so on. However no proof is given here, it's worth noting that these *levels are closely coupled with the levels of generative grammars*.)

Rule set

A huge company of systems can be described so that individual descriptions require only a relatively small "room" (e.g., a shorter series of symbols), if they can be specified as a kind of definiteness.

Consider the following method: Take some comparable systems and construct some mappings $c_i \mapsto \sigma_j$, where c_i (called the **rule condition**) and σ_j (the **rule consequence**) are both the positionings of a given concept of a comparable system (and are, therefore, comparable to those systems). The interpretation of the mappings is as follows: If a system satisfies the condition c_i, that is, if c_i is an abstraction of the system, then the consequence σ_j is also satisfied by the system, yielding that it will also be an abstraction of the system (and therefore an information regarding the system, unless σ_j is the arbitrary abstract system). Systems c_i and σ_j may be constructed using certain arguments, therefore by specifying arguments *p*, the **rule** $r(a_{p\in P})$ produces a mapping $c_i \mapsto \sigma_j$.

When specifying rules, let's tag those **actual rules** for which it stands that if the left sides of the mappings related to some actual rules are satisfied, then the right sides shall be collectively satisfied, too, and therefore those rules are not contradictory. In addition, the consequences of actual rules are required to not be the arbitrarity. Not necessarily actual rules are called **abstract rules**. The aggregate of (parametric) rules, when actual rules are tagged, is called a **rule set**.

If the right side of a rule is an arbitrary abstract system, the rule does not constrain behavior (and therefore the rule is not an actual rule), and that's why rule sets may be extended arbitrarily with such rules and why such rules can be omitted. Rules of this kind are called **arbitrary abstract rules**.

Rules define a simple deduction: *"if"* the left side is satisfied, *"then"* so is the right side. For a satisfying rule, however, both the left and the right side define a consequence of the system, which are collectively satisfied in this case, i.e., the conjunction of the two is also a consequence of the system. For this reason, satisfying but non arbitrary abstract rules are indeed actual information regarding the system. According to the above-mentioned statements, applying a rule is actually the recognition of a situation, the appearance of a constrainer, where the appearance of the requirement or condition part of the constrainer implies the existence of the consequence and the entire rule as constrainers.

(*An actual rule is information regarding the system*, if neither sides of it is the arbitrary abstract system.)

(***Rules are concepts of the system, while actual rules are actual concepts of the system.***)

Applying a rule, therefore, is not like the traditional concept of logical implication, but a kind of deduction, i.e., a new statement resulting from combining rules with each other. Applying a satisfying rule is actually the recognition of a concept "appearing" at a given location and time.

Actual rules loan some kind of *orderliness* to the system.

For constrainer rules of the form $\mapsto \sigma_j$ (that is, for unconditional rules), this orderliness means that the system must choose a state to rest in from a proper subset of its possible states at the defined moments of constrainer σ_j of the system (rather than having the option to choose from all of its states).

Orderliness appears between constrainers c_i and σ_j in case of rules defining mappings $c_i \mapsto \sigma_j$, meaning that satisfaction of c_i or, in other words, the appearance of concept c_i implies the appearance of σ_j or, more exactly, the collective appearance of c_i and σ_j.

Orderliness has special significance when appearing repeatedly in a system, for example, in another moment or at another location. In case of multiple appearances, repeated appearances can be extracted, and then actual forms of appearances can be substituted by applying rules.

Extraction may decrease the size of representation once the extracted rule and its applications can be represented in a shorter form (e.g., with less symbols) than the actual behaviors. And this is exactly the case when it's practical to perform extraction.

(Rules describe relationships of the *"if-then"* kind, much like impacts between entities do. Rule is a generalization of the idea of impacts, enabling the condition and the consequence to be located in different abstraction levels. ***To put it together, impacts within a system are actual rules of that system.*** Rules, furthermore, allow for parameterizing, and the sides of the rules are defined as the positionings of certain concepts.)

Intelligence as Capability of Understanding and Abstraction

Our approach basically discusses the apparatus of representing systems and refining knowledge regarding systems. Accepting hypothetically or only as a possible approach that "intelligent behavior" is based on awareness or recognition of situations or, in other words, on the acquisition of knowledge regarding the situations, the apparatus of the approach can be employed to think about the nature of intelligence. Emphasizing again, the goal is not to give a complete definition of intelligence, but only to think further about an *opinion* that may help in the detailed analysis of the nature of intelligence some time in the future.

Assume the followings as our hypothesis:

Intelligence *is the capability to identify externally specified rules or repetitions in changes, that is, the capability to identify orderliness or patterns.*

Based on this assumption, *an observer can be said to have* ***higher intelligence*** *than the others if that observer can discover or recognize significantly more actual patterns regarding some given systems (i.e., a given category).*

Now let's analyze the components of the above phrasings:

- Intelligence can be interpreted comparatively, relatively to other observers.
- The degree of intelligence (that is, *higher* intelligence) is interpreted only for given systems or categories, enabling different relations of intelligence of observers for other categories.

- Multiple patterns need to be identified. One way to achieve this by the observer of higher intelligence is the identification of all patterns identified by the others, and identifying some more patterns in addition. The weighted means of the identified rules form an ideal measurement for comparison even in mixed cases.
- It is of high importance that the concept is about identifying actual patterns. For example, identification of a false impact (called superstition) appearing in a non-domain abstraction does not increase intelligence. Weighting can also be employed when determining the importance or relevance of actual rules and, in general, rules.

The above wordings are indeed definitions and were labeled hypotheses only because intelligence is a widely used term both in everyday's life, both as a psychological term. Hypothesis, in this case, is a working theory, and has the goal to help to discover some features of "real" intelligence.

Now let's see what features can be derived from the definitions. Do not forget that "higher intelligence" regards only a given scope or category of concepts.

Problem resolution. Intelligence is basically defined only as the capability to be aware of (or recognizing) situations. This is, however, closely coupled with problem resolution. Problem resolution is, on the one hand, the identification of rules regarding a situation, especially the condition parts of those rules. It can also include the combination of rules with each other, that is, the act of deduction, however collective interpretation and application of rules is not necessarily deduction. On the other hand, problem resolution covers the application of the resolution method found, but this does not belong to intelligence, since others may also implement or execute a resolution method found by someone else.

Intelligence in itself is basically passive or, to amend, its "activity" is only the recognition of situations, where passivity is used in the sense that the concept of intelligence is employed only to identify situations, that does not include any actual intervention to the situation or any action in itself.

Abstraction and concept construction (ideation). Higher intelligence identifies repetitions and patterns appearing in the system better and, by extracting, treats as individual concepts. Recognizing and extracting repetitions and patterns in a system is actually the act of abstraction.

Higher intelligence identifies unexpected situations better, since either it can apply its earlier concepts to those situation, or it can construct new concepts from the patterns of the new situation, and therefore it can determine the internal relations of the new situation.

Higher intelligence impenetrate deeper the semantics of its extended context. Considering the observer as an entity of the system, it can not recognize more than the change of itself and its context, collectively called the extended context.. Recognition of changes usually happens via an interface (signal channel) and by means of signals. By considering the signals as mappings of the changes of the external context, the observer can identify the mapping method (or mapping rules), as well as the patterns of the changes serving as the source of the mappings. It comes to say that the observer can, up to its capability to identify patterns (analogy, cohesion, structure etc.) and repetitions, uncover or interpret the meaning of the signals described in a given form with a given "syntax", as well as the patterns or the semantics (meanings) of the signals.

Cognizance. Based on our speculations, higher intelligence has more knowledge about the patterns of the changes of its context and can cognize it better.

Higher intelligence may be able to describe identical changes using less room or in shorter form (for example, using less symbols), since it is capable to extract repetitions and patterns appearing in those.

Reverting the previous rule, *higher intelligence is able to represent more or more complex changes using the same room*.

More intelligent representation method. Now we have enough bits to construct the idea of more intelligent method of representation, actually as a form of specification or definition, in which descriptions are less redundant due to analogy extraction, and therefore descriptions are basically of smaller size.

Intelligent behavior. Were intelligence based on deduction, it were to come to ask whether a program can be considered intelligent if it does not

combine but only applies rules? Particularly, programs showing appropriate behavior in more (essential) situations are considered more intelligent. Similarly, a human being can behave "appropriately" in a given context without the need to deduct consequences or combine rules, exclusively based on its background (raising) and his/her character.

Our approach of intelligence implies that a program can be considered more intelligent if its behavior is more conformable in a situation or, in other words, it can more accurately determine the nature of certain situations and it can apply the rules of the necessary "conformable" behavior to those situations.

Survival and intelligence. Our definition given for intelligence may be related to survival and adaptiveness. *Critically low* intelligence is not capable to identify situations threatening survival, nor the methods to resolve those, and therefore it is certainly deteriorative in the aspect of evolution. Environmental adaptation, that is, the construction of suitable rules requires identification of environmental rules.

Survival, moreover, requires some kind of activity, too. According to a previous statement, however, intelligence is basically passive and its activity is limited to determining situations and the rules to apply to situations, but does not necessarily include the application of the rules.

Try to imagine "critically high" intelligence. Since critically high intelligence performs better in identifying repetitive concepts and rules, this identification process may employ its entire behavioral resources, practically not leaving enough resources for performing actions. The identification of rules and possibilities, called the identification of possible rectification methods, causes it to continuously confront the imperfection and incapacity of its environment, and so its own limits, further causing inability to realize even its own, otherwise proliferating ideas. (A great example for this can be seen in the movie title Phenomenon 1977, directed by Jon Turteltaub, written by Gerald Di Pego and starring John Travolta, Kyra Sedgwick, Forest Whitaker and Robert Duvall.)

Allowing for some tautology, it can be stated that only utilizable recognitions are useful. An abstract and unrealized recognition is therefore unfortunately non-useful, and is unnecessary in this sense until an unexpected or random situation, however, establishes its necessity.

Critically high intelligence is accordingly deteriorative to evolution, and this may be the reason why evolution has fixed it at about the current level measured by IQ tests as a value of cc 100. By blurring and glazing the imperfection of the world and by hiding many of the possibilities to mend it, evolution ensures we can pay enough attention to deal with one or the other bits of the world.

It's important to note that direct evolutional benefit is not the only and not the best measurement to judge in the questions of usefulness and of the behavior of intelligence. Individuals and companies of individuals both employ complex and multileveled "preference functions" to determine usefulness, where, in addition to the primary considerations of survival, internal quality of life, the excitement of discovery and creation, and the experience of beauty and harmony play major roles.

(MoNo)

Modeling Notation

Our theoretical approach can help in refining notations used in modeling languages. This can lead to the development of the tools of—being itself a paradox—a generic DSL (generic domain-specific language), which would be sufficient to specify the basic information about the structure and behavior of any system.

The followings show the basic elements and concepts of an experimental modeling notation (referred to as MoNo). Notations are usually identical to those of the well-known UML (and OMT). MoNo, however:

- Offers both visual and textual specification options (visual manifesting in diagrams, while textual meaning some kind of structured form, such as XML, or there is even a textual specification option resembling written English language).
- Comes with only one type of diagram on which all diagrams of UML can be drawn and, of course, a system may be represented in multiple diagrams, spotting or capturing different parts of the systems and showing those parts in different granularities.
- Representations of a system with more or less details (i.e., at higher or lower granularity) can be generated automatically.
- Backward-compatible with UML's apparatus.
- Features exact "abstract semantics", that is, by choosing the platform and making design decisions, the software application can be created via generation. In other words, MoNo is not only able to represent the blueprints or drafts of a system, but it can also be considered and used as a programming language.

The following sections introduce the most fundamental visual notations of MoNo.

MoNo has four base elements:

- Structural elements, the building blocks of systems, basically represented by rectangles.
- Changes in time (“durations”) basically reflected by rectangles with elliptic sides (elliptic rectangles, for short). This is, for example, the notation of state and activity (the two being similar concepts: the system rests in a given state until an activity spanning over some time is being executed, such as while "having a lunch", we are in the "having lunch" state).
- Elements related to moments and for which time (duration) is ignored, shown by ellipses. Therefore, this is the notation of actions, the uninterruptible operations, but it is also appropriate for denoting sets of operations (i.e., interfaces).
- Descriptor or informative elements, used to specify details and add notes.

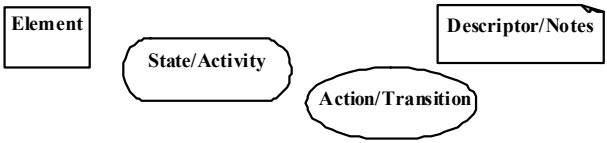


Figure 22: MoNo's base elements

Generally speaking, appearance of elements can be substituted by small pictures or icons.

The upper compartment is called the *name compartment*, followed by the definition compartment, which may contain textual and graphical (sub-) compartments, even alternating between those. Should the graphical compartment grow big, element data can be shown in a separate shape placed on the top left corner (as is in the next figure in the case of context).

Types may be *abstract*, written in italics. Elements within an element may be specific (such as a property or a compartment), shown by underline. Elements, furthermore, may be *multiple*, shown by two small shapes put on top of each other with some offset.

Elements may also be *referenced*, shown by dashed lines (this is suitable, for example, to refer to an event or action occurring in the external word).

Finally, elements can be *global* (similar to the "class attribute" and "class method" of UML), shown by double underlines or with an initial $ symbol in the name.

Structural elements are basically specified in rectangles. This is the notation of entity types ("*classes*") and entities ("*objects*"). A single entity ("object") can be defined on its own, without specifying its type ("class").

The valued nature of an element (in case of value elements, such as a composite value) can be emphasized by parallelogram notation.

The external contributor nature of an element (such as in case of a user) can be emphasized by stickman notation.

"Context" can be represented by double-sided rectangles. Context is, usually, an element having its own properties and operations, which it can access. (The package and component elements of UML can be defined as context.)

The nature of being a "physical device" (think of a computer, for example) can be emphasized by "cube" notation.

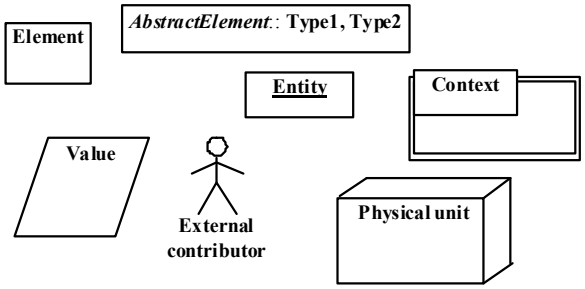


Figure 22:Structural base elements

Structural elements can be connected several ways.

Solid (non-dashed) lines denote relation or association. The ends of the relations are the roles, for which multiplicity can be specified.

- Multiplicity can be defined in the form of *mNumber* or *mNumber1..mNumber2*.
- Natural numbers include the * or ● symbol of "arbitrarily many"; itself being the notation of 0..* multiplicity.

- Multiplicity can also be indicated by "multiplied shapes" (that is, multiple instances of the same shape placed on top of each other with some offset for visual overlapping).
- The *0*, ○ and *?* symbol indicates non-mandatory (optional) nature.
- The 1 and ▌ symbol indicates mandatory nature. (Based on the above notations, a ●▌ symbol placed at the end of a line is the symbol of the *"at least one, but even arbitrarily many"* (1..*) multiplicity).

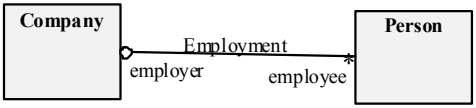


Figure 23: Relation, role and multiplicity

Lines ending in triangles denote generalization-specialization (resembling the ⇒ symbol of the implication ("if-then" relationship) concept of logic). Multiplicity can be specified both at the general and specialized end of the association, meaning that the more general and the more specific elements can be accessed individually as different elements, such as in the case of (the relation of) a *Product Description* and a *Product*.

The compartmental "whole versus part" relation (*"aggregation"* and *"composition"*) can be represented by placing an empty or a filled diamond, respectively, to that role of the relation, which represents the whole in the whole-part relation.

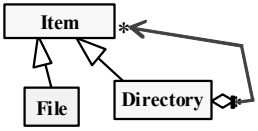


Figure 24: Specialization and aggregation

A relationship may have its own properties and operations (and, moreover, even its own relations etc. like the *"association class"* in UML). Such relations are represented by hexagons (with two apexes on the two sides). Relationships may be multiple (e.g., the same Person may be in Employment relation with the same Company in several different ways).

Data also can be attached to roles, therefore roles can also appear as special entities (being special in the sense that such an entity is related to that base entity which it is a role of). Roles may be shown at the side of the base entity (and therefore this notation is not the same as that of UML's "qualifier" notation), or the role can be connected to the base entity with a dashed line ending in a triangle (and therefore this is not the same as the notation of "realization" in UML).

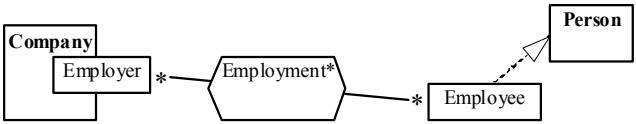


Figure 25: Individual notations of (multiple) relations and roles

The two base elements related to change are the notations of *process* and *transition*.

- *Processes* can be coupled with *durations*, therefore process is the notation of *states,* and, as well, of multi-step interruptible operations, called *activities*. The visual notation is the rectangle with elliptic sides. (*Long-term processes*—such as a long-term collaboration—appear like entities and are, for example, persistable.)
- *Transition* is the notation of uninterruptible operations related to certain moments, and of those operations for which time is ignored (e.g. an action or a transaction). Its visual notation is the ellipse.

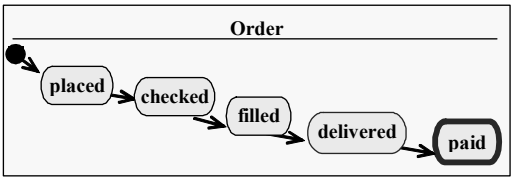


Figure 26: States and state changes of an element

Change can be specified within structural elements or with lines connecting to the sides of structural elements.

- *Public* change elements are written in bold (public change elements represent things able to be set externally in case of states and able to be called externally in case of operations).
- *Generalization-specialization* relations may be defined between change elements of the same type (denoted with a line with triangle head), and that relation may be abstract (written in italics).
- A state within another state represents a (specialized) substate. (A structural element is inherently a notation of its most general state ("state of existence") as well).
- Additional operations can be defined within an operation for specifying operations interpreted only in that state (or interpreted in a given way in that state).

    - Internal operations can be specified on the side of the outer operation or as connected to the outer operation with a line.
    - Arrow-headed lines originating from a structural or operational element serve as shorthand for representing operations triggering transitions.

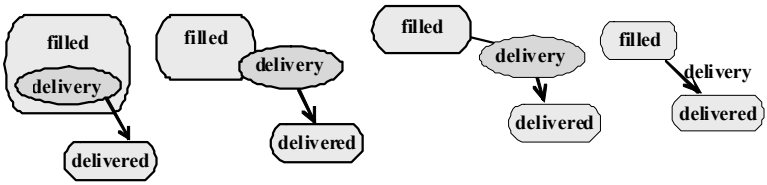


Figure 27: Notations of a transition from a state triggered by an event

The notations are backward-compatible with the lifeline and swimlane notations of UML. UML's lifeline actually represents a series of operations related to the given structural element. With the swimlane notation, these operations are shown within the given element (as the lane actually represents the container element).

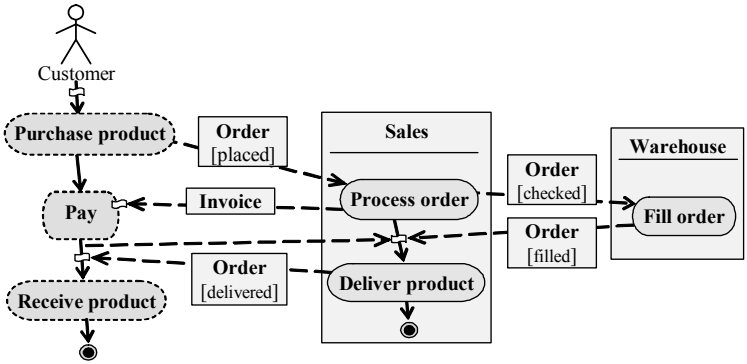


Figure 28: Operations

The shape with dashed edges denotes activities or events of the external world, or tasks on which the system is notified (via, for example, a task manager).

The wavy shape (or possibly just a ~ symbol) represents delay (optionality or synchronization), that is, the step or the sending of the affected message may not happen immediately or at all. A message received by the wavy shape becomes the condition of the transition or operation (synchronization).

Sets of operations can be better emphasized by double-sided ellipses. This allows for representing interfaces (resembling to UML's lollipop notation) and collaborations. For easier drawing, these can be drawn with the base shape. In more detailed representations these can also specify defined internal operations (either visually or textually).

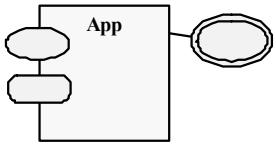


Figure 29: Externally accessible operations and interface of an element

Elements denoting operations can also be connected:

- Anonymous (normal) arrows denote the next operation of the execution. The wavy icon of delay (optionality/synchronization) indicates that the step is performed only after some time (when some given external conditions are met), or is performed optionally.
- Anonymous *dashed* lines originating from an operation or transition toward an operation or transition:
    - „Normal" arrowhead indicates asynchronous messages or the ignorance of call return ("trigger" calls). Should such an arrow point to a transition, sending the message triggers the transition. (If the destination element is in an inappropriate state, an error occurs.) If the arrow points to the wavy icon of delay, the message is the condition of the transition, message or operation.
    - Filled arrowhead denotes synchronous messages, the "calling" of the operation or transition.

The message name shown on the message transfer's visualization tells the name of the operation to invoke on the destination element. Both multiplicity and optionality can be shown on transitions and messages.

The rule of thumb is that non-directed dashed lines (called "linking") connects additional elements related to the source element.

The notations enable to specify use case diagrams used in UML.

- Use cases in UML are indeed externally accessible operations of the system, and the duration of those operations is ignored. Therefore, use cases can be represented as the operations of the system specified as a structural element.
- The same notation can be used to represent operations of services (such as web services) and the relationships of systems or services.
- *Use cases* of an actor can be denoted by a dashed line originating from the actor and ending in an arrowhead at the operation. Delay can be shown on the lines (with the wavy icon) in more detailed diagrams.

- The *"include"* concept of UML can be represented by a dashed line with filled arrowhead originating from the base case (the one which includes another).
- The notation of the *"extend"* concept of UML (reversed to how UML denotes it) can be represented by a dashed line with normal arrowhead originating from the base case, and by possibly indicating optionality.

The act of sending an event can be denoted with a separate shape, to which the data to be sent can be connected. Event reception can be specified either as a simple operation (action) or as a separate shape. The latter is semantically identical to a synchronous call.

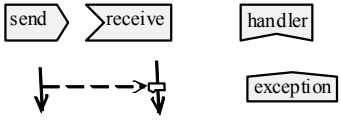


Figure 30: Notations of event generation/reception and exception handling

The notations of send and receive are suitable to specify the input and output of a complex calculation or algorithm (and also to specify the sent and received parameters in an operation call).

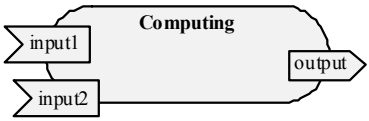


Figure 31: Computation

The above notations can be used to describe generic systems (including business, software and other kinds of systems). Finally let's consider some examples for the notation.

In our example, there is some kind of relationship between Customers and Suppliers:

Figure 32: Relationship between Customers and Suppliers

This relation is manifested in the (execution) process of the activity ("collaboration") called Ordering:

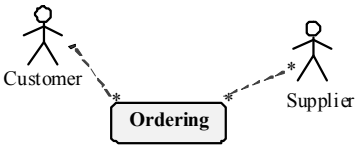


Figure 33: Ordering is a common (shared) activity

The operations of the collaboration (think of "use cases" and "functionality") can also be specified:

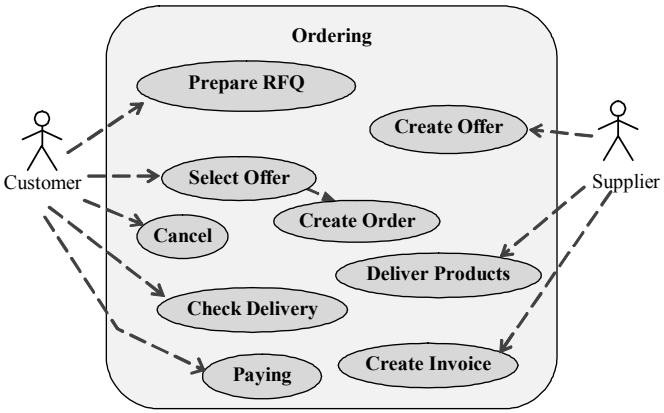


Figure 34: Operations ("functionality")

Individual functions can be specified in more details (think of UML's *"include"* and *"extend"*):

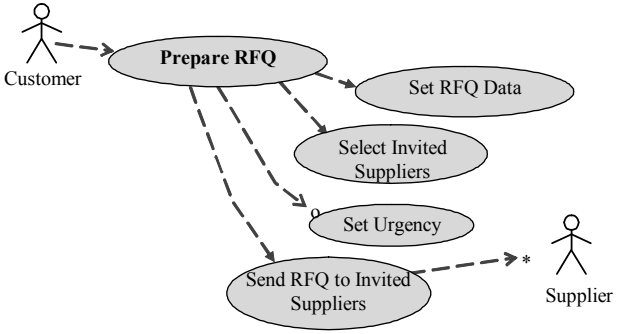
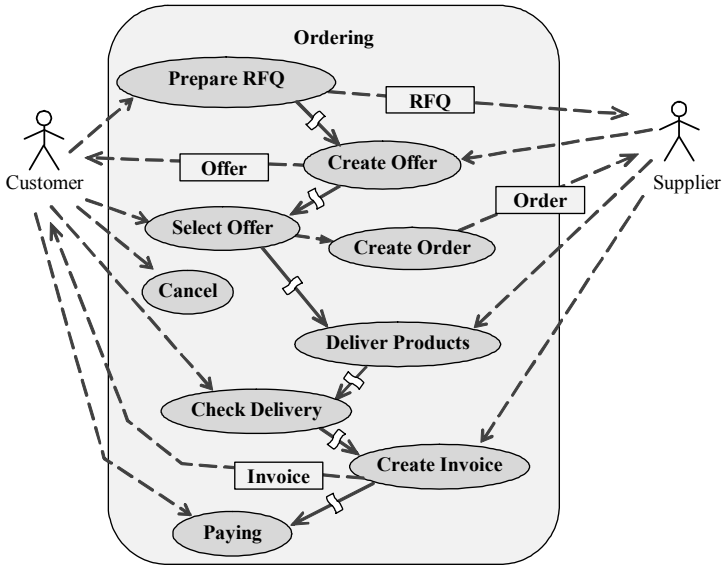


Figure 35: Specifying an operation's details

Timelines (or timing) and the sent/received data also can be defined:



36. Figure 37: Operations ("functionality")

The model can be refined further by specifying states (and the operations to be executed in the states). You can also show the series of activities ("lifelines") of the contributors or participants, along with the order of the optional or mandatory tasks they have to execute.

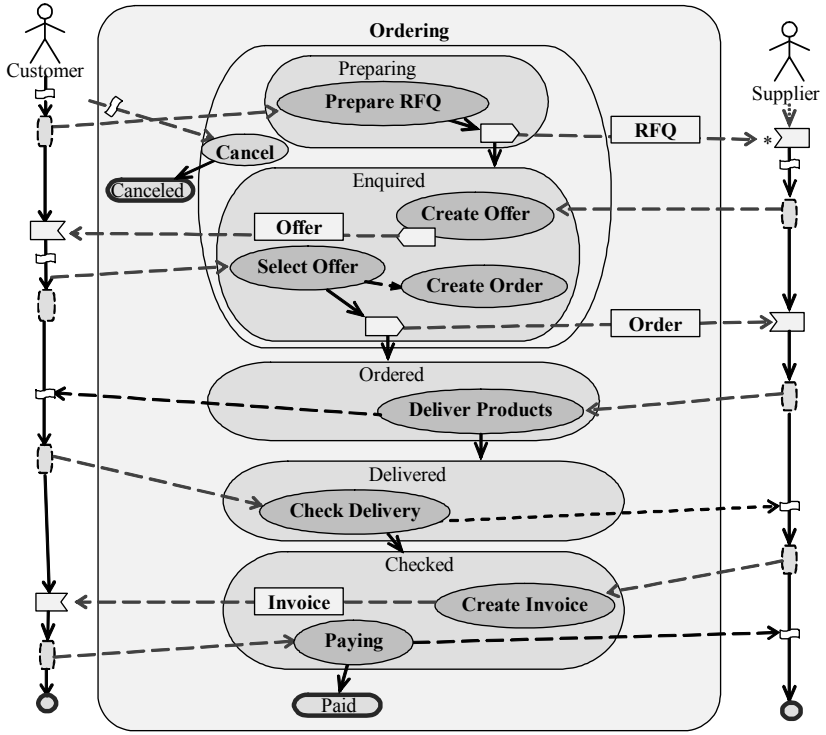


Figure 38: Operations ("functionality")

The model can also be specified with structured text (such as XML) and with a special language resembling English. This experimental language currently uses 15 sentences/subclauses and 11 "operand" forms. For example, *"a/an*" introduces a formal parameter, *"the"* introduces actual parameters, while the ~ symbol is the short for the English "it" (the *this* keyword in C++) and so on.

These structures allow for specifying structural relations, states, state transitions, contributors (participants), (mandatorily) executable tasks, control structures etc. The appropriate program or engine is capable of executing and tracking the specified processes. The following can be a draft for the example described above:

```
an ordering of a customer:
  ~ may have an RFQ
  ~ may have invited supplier participants
  ~ may have offers
  ~ may have a selected offer
  ~ may have a supplier participant what is the supplier of the selected offer
  ~ may have an order

  ~ can be invited
    the customer can cancel the ordering
      ~ is canceled
    preparing
      create an RFQ of the ordering
      the customer can set the RFQ data
      the customer can add invited suppliers
      the customer can remove invited suppliers
      when the customer finish preparing
        ~ is requested for a quote
    ~ can be requested for a quote
      firstly
        for each invited supplier of the RFQ
          send the RFQ to the invited supplier
          the invited supplier may create an offer
      when an offer is received from a supplier
        it is an offer of the ordering
      when the customer creates an order of an offer
        the selected offer of the ordering is the offer
        send the order for the supplier
        ~ is ordered
  ~ can be ordered
    the supplier must deliver
    the customer must receive the delivery
      ~ is delivered
  ~ can be delivered
    the customer must check the delivery
      ~ is checked
  ~ can be checked
    the supplier must create an invoice
    the customer must pay
      ~ is  paid
  ~ can be paid
  ~ can be canceled
```

Some sections of the XML transcript:

```
<entity name="ordering" of="customer">
  <has name="RFQ" mandatory="false"/>
  <has name="invited supplier participant" multiple="true"/>
  <has name="offer" mandatory="false" multiple="true"/>
  <has name="RFQ" mandatory="false" />
  <has name="selected offer" mandatory="false">
  <has name="supplier participant" value="selected offer. supplier"/>
  <has name="order" mandatory="false">

  <state name="invited">
    <task name="cancel" mandatory="false">
      <when name="cancel">
        <set name="cancel"/>
       </when>
    </task>
  </state>
...
  <state name="ordered">
    <task for="supplier" name="deliver" mandatory="true"/>
      <when name="delivered">
        <set name="delivered"/>
      </when>
     </task>
   </state>
  <state name="delivered">
    <task for="customer" name="check the delivery" mandatory="true"/>
      <when name="OK">
        <set name="checked"/>
      </when>
      ...
     </task>
   </state>
  <state name="checked">
    <task for="supplier" name="create invoice" mandatory="true"/>
      <when name="OK"><where name="invoice"/>
        <send name="invoice" to="customer"/>
      </when>
    </task>
    <task for="customer" name="pay" mandatory="true"/>
      <when name="OK">
        <set name="payed"/>
      </when>
     </task>
   </state>
  <state name="paid"/>
  <state name="canceled"/>
</entity>
```

VÉG CSABA

INFORMÁCIÓTUDOMÁNY
Áttekintés

Az informatikát, azon belül is az információs rendszerek fejlesztését tekinthetjük egy olyan katalizátornak, amely problémáival, illetve eszközkészletével segít kialakítani egy tiszta fogalomrendszert az információ ábrázolásáról, illetve az ismeretszerzésről. A cél egy olyan elvi megközelítés felvázolása, amely egyszer elvezethet a rendszerfejlesztés egzakt fogalmi kereteinek kialakításához, valamint az általános értelemben vett ismeretszerzés és reprezentáció legalapvetőbb fogalmainak matematikai precizitású tisztázásához.

Információtudomány

Elméleti megközelítés

A „Feljegyzések…”[1] c. könyv egy kezdeményezés az információtudomány legalapvetőbb fogalmainak a meghatározásához. A következőkben ennek a kezdeményezésnek a legfontosabb elemeiről találunk egy áttekintést. A pontos matematikai megfogalmazások megtalálhatók az eredeti könyvben.

Milyen gyakorlati előnyökkel járhat egy elméleti megközelítés kidolgozása a rendszerfejlesztés, illetve az informatika témaköreiben? Az informatikát most a *rendszerekre vonatkozó információk megszerzéseként, ábrázolásaként és átalakításaként* definiáljuk, melybe a programrendszerként, más néven „alkalmazásként” történő reprezentációt is beleértjük. Milyen eredményeket hozhat egy ilyen fogalmi rendszer kialakítása?

Elvi oldalról tekintve, egy megfelelő elvi megközelítés segítségével pontosíthatjuk az információtudomány, illetve részben az informatika vizsgálati területét, mely alapvető témaköreit az információk megszerzéseként, ábrázolásaként és átalakításaként határozzuk meg. Például egy informatikai rendszer fejlesztése jelenti egyrészt a követelmények meghatározását (ami ismeretszerzés), majd azok ábrázolását vázlatos (pl. elemzési, tervezési) modellekként, illetve programként; valamint jelenti ezen ábrázolások átalakításait, módosításait is. Maga a rendszer működése is értelmezhető a külső eseményekről, adatokról (a felhasználói felületeken keresztül) történő ismeretszerzésként, az ismeretek ábrázolásaiként és átalakításaiként. Egzakt fogalomrendszer esetén pontosan le tudjuk írni, hogy munkája során éppen mit csinál egy informatikus, vagy, hogy mit jelent egy rendszer adott működése.

[1] Vég Csaba: Feljegyzések az információtudomány elméleti alapjairól (2008. http://mek.oszk.hu/06300/06303)

Mivel a fogalomrendszer általános értelemben tárgyalja az ismeret fogalmát, annak ábrázolását és átalakításait, ezért az eredmények alkalmazhatók informatikán kívüli területekre is (mérések, kísérletek, hétköznapi események, stb.).

Gyakorlati, technológiai szempontból tekintve az információ ábrázolásával kapcsolatos elvi alapok lehetővé tennék magas szintű jelölésrendszerek kialakítását. A jelenlegi „magas szintű" programozási nyelvek, és a magas szintűnek tartott modellező jelölések (pl. OMT, UML, ER diagramok, stb.), mind a számítógép gépi logikájából fejlődtek ki ún. „bottom-up" módon. A fejlesztéseket az vezérelte, hogy adott jellegű utasítás-sorozatokat egyszerűbben meg lehessen adni. Így ez a fejlődés esetleges: adott technikák megvalósultak, adottak viszont nem. Megfelelő elvi alapok esetén a jelölésrendszereket „top-down" módon lehet kialakítani, illetve meg lehet határozni a szükséges és elégséges (!) jelölések körét. Ez elvezethet „információs nyelvekhez", amelyek megközelítik az emberi (technikai) nyelv kifejezőerejét. Ezt kb. úgy képzelhetjük el, hogy egy megfelelő szervezési dokumentáció azonnal működtethető rendszert eredményezne. Számszerűleg mérve ezek 1-2 nagyságrenddel magasabb absztrakciós szintet jelentenek, azaz ilyen nyelvek esetén egyetlen sornak 20-200 sornyi „magas szintű" program felel meg.

Egy elméleti megközelítés ugyancsak lehetővé tenné olyan különleges „információs rendszerek" kialakítását, amelyekkel emberi nyelvhez hasonló módon (akár szóban) közölhetünk adatokat, eseményeket, vagy akár a rendszer működésére vonatkozó szabályokat, s a rendszer automatikusan módosítja a belső állapotait vagy épp szabályait, megkeresve az információval kapcsolatos definíciós helyeket. Ilyen rendszerek egyszerűbb változatai a reagálás/együttműködés alapú leírások és rendszerek, amelyek részei „intelligens" (de természetesen programozott) módon követhetik a külvilág eseményeit, reagálhatnak arra, illetve együttműködhetnek további rendszerekkel. Ez a módszer az üzleti alkalmazások definícióját is leegyszerűsíti, de pl. robotvezérléseknél is alkalmazható.

A megfelelő elvi fogalomrendszer arra is alkalmas, hogy általános célra szánt frameworköknek, generátoroknak, vagy akár jelöléseknek meghatározza a használhatósági körét.

I.

Rendszer, absztrakció és kompozíció

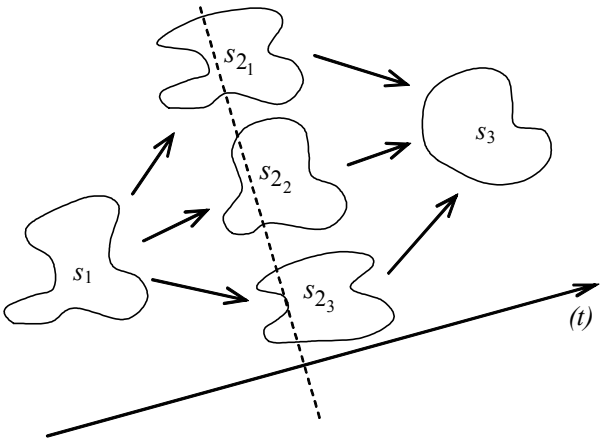
Rendszer

Az információ-tudomány alapvető céljaként az információ megszerzésének, ábrázolásainak és átalakításainak a vizsgálatát határoztuk meg. Információ azonban mindig valami*re* vonatkozik, s ezt a „valamit" a megközelítésünkben *rendszernek* nevezzük. A rendszer egyetlen megfigyelhető tulajdonságának az *állapotát* és annak *változását* tartjuk; a rendszer így valójában egy *folyamatként* jelenik meg.

A következőkben bemutatott megközelítés egyik újdonsága, hogy (általános megadású rendszer esetén) azonos módon kezeljük a rendszert és annak *megfigyelt* változatait. A megfigyelés *bizonytalanságként* jelenik meg, azaz adott időpontokban csak az állapotok egy-egy halmazát tudjuk meghatározni (pl. a hőmérséklet éppen 36 és 37 fok közé esik, de a pontos értéket nem ismerjük). Megközelítésünk másik újdonsága, hogy a rendszert *teljességében* kezeljük: az állapot a teljes rendszerre vonatkozik, bár az állapotok lehetnek összetettek is.

Egy $\sigma = (\mathcal{T}, S, P)$ (**általános megadású**) **rendszer** esetén így megadjuk az időpontokat (azok nem üres, rendezett halmazát: $\mathcal{T}$), a lehetséges állapotokat (azok nem üres halmazát: S), és a folyamat $P : \mathcal{T} \to 2^S \setminus \{\varnothing\}$ függvényét, ami minden időponthoz egy (nem üres) állapothalmazt, egy állapotkombinációt rendel (adott időpontban a rendszer milyen állapotokban lehet). Ez a definíció mindössze annyit jelent, hogy a rendszer egy adott időpontban több állapotban is lehet. Nem tudjuk pontosan, hogy éppen melyikben van, csak az állapotok egy részhalmazát, azaz egy kombinációját tudjuk megadni. Ha egy időpontban semmit sem tudunk az állapotról, akkor ott a teljes

állapothalmazt vesszük fel ($P(t)=S$), s ekkor a rendszer **bizonytalan**, **tetszőleges állapotú**. A többi (nem tetszőleges állapotú) időpontot együtt a **rendszer idejének** nevezzük (jelölés: T, ahol $T \subseteq \mathcal{T}$). Az időpontok eredeti $\mathcal{T}$ halmaza valójában a megfigyelő idejét jelöli.



39. ábra. Általános megadás esetén a rendszernek egy időpontban több állapota is lehetséges

Bizonyos vizsgálatok egyszerűbbek, ha a rendszert a $\sigma = (\mathcal{T}, T, S, P)$ formában, **determinisztikus megadással** definiáljuk, ahol az idő és a lehetséges állapotok mellett a rendszer idejét is felvesszük, a változás függvényét pedig olyan $P : T \to S$ formában adjuk meg, amely minden ($t \in T \subseteq \mathcal{T}$) időpontban a rendszer által felvett (pontosan egy) állapotot határozza meg.

Ha adott $\sigma_{i \in I} = (\mathcal{T}, S, P_i)$ rendszerek idő- és állapothalmaza azonos, akkor azokat **közvetlenül hasonlíthatóaknak** nevezzük (jelölés: $\Sigma^{(\sigma_i)}$, ill. a σ rendszerrel közvetlenül hasonlítható rendszerek jelölése: $\Sigma^{(\sigma)}$), mivel ekkor az időpontokban megvizsgálható, hogy a rendszerek által felvett állapotkombinációk milyen viszonyban vannak (pl. melyik a bővebb).

Lehetséges, hogy egy rendszer minden időpontban bizonytalan állapotú, amit ekkor **tetszőleges absztrakt rendszernek** nevezünk és $\aleph$ módon jelölünk.

Két (σ és σ') rendszer esetén lehetséges, hogy különböznek az idő- és állapothalmazaik, de ugyanolyan *jellegű* módon változnak, azaz létezik a rendszer idejére és állapotkombinációira is egy-egy ($\alpha_{\equiv_T} : T \to T'$ és $\alpha_{\equiv_S} : \overline{S} \to \overline{S}'$) invertálható (ill. idő esetén ugyanolyan sorrendű változást eredményező, azaz szigorúan monoton) függvény, hogy az (esetlegesen megfigyelt) változás állapotkombinációinak a részhalmaz-viszonyait, azaz a *változás lényegét* nem módosítja. Ekkor azt mondjuk, hogy a két rendszer **ekvivalens** ($\sigma \equiv \sigma'$), vagy más néven, az egyik rendszer a másik **rendszer átjelölése**. (Celsius fokokban mért hőmérsékletváltozásnak, mint rendszernek az átjelölése, ha az értékeket Fahrenheit vagy Kelvin fokokban adjuk meg. Ezek a rendszerek ekvivalensek.)

(Minden tetszőleges absztrakt rendszer ekvivalens.)

Az időpontok és az állapotkombinációk konverziójánál használt ($\alpha_{\equiv} = (\alpha_{\equiv_T}, \alpha_{\equiv_S})$) függvény-párost rögzíthetjük, s ekkor azt **közvetlen átjelölésnek** nevezzük. Egy közvetlen átjelölés azonos idő és állapothalmazú (azaz közvetlenül hasonlítható) rendszerek két csoportja között teremt meg kapcsolatot, azaz egy-egy rendszernek megadja a vele ekvivalens, tehát azonos jellegűen változó átjelölését.

A rendszer állapotai lehetnek összetettek is, azaz előállhatnak elemibb állapothalmazok (általánosított Descartes-féle) szorzatából. Ez alapján definiálható az azonos $\mathcal{T}$ időn értelmezett **rendszerek kompozíciója** úgy, hogy a kiindulási rendszereket részeknek tekintjük és meghatározzuk, hogy melyik rész milyen helyre vonatkozik. A **részeket komponenseknek** vagy **alrendszereknek** is nevezzük.

A rendszer (azaz a folyamat) *lényegében* nem változik, ha a $\mathcal{T}$ időhalmazt nem értelmezett időpontokkal felbővítjük, vagy azokat elhagyjuk. Hasonlóan, egy kompozit rendszer *lényegében* ugyanaz marad, ha adott időpontokban értelmezetlen állapotú részeket hozzá veszünk, vagy ilyeneket elhagyunk. Az ilyen, „lényegében ugyanolyan" rendszereket **azonosnak tekinthető rendszereknek** nevezzük (jelölés: $\sigma \Leftrightarrow \sigma'$; a σ-val azonosnak tekintett rendszer: $\sigma^{\Leftrightarrow}$).

(Az azonosnak tekintett rendszerek egyben ekvivalensek is. A tetszőleges absztrakt rendszerek azonosnak tekinthetők).

Hasonlítható rendszerek

Ha a rendszerekhez értelmezetlen állapotú időpontokat vagy részeket veszünk fel, vagy ilyeneket elhagyunk, akkor lehetséges, hogy a rendszereknek azonos idő- és állapothalmazaik lesznek. Az ilyen rendszerek így (közvetlenül) összehasonlíthatóvá válnak, ezért ezeket (közvetve) **hasonlítható rendszereknek** nevezzük (jelölés: $\Sigma^{(\sigma_i)^*}$, illetve a σ-val hasonlítható rendszerek halmaza: $\Sigma^{(\sigma)^*}$).

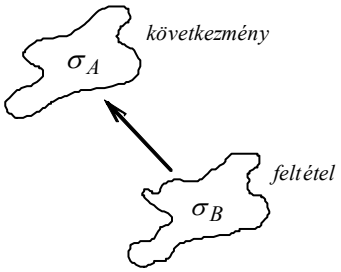
(Egy kompozíció és a részei hasonlíthatók.)

Hasonlítható rendszereknek az adott időpontban felvett állapotkombinációik, így ezáltal az állapotváltások teljes folyamatai is összehasonlíthatók, valamint a rendszerekkel további műveletek végezhetők.

Hogyan történhet az összehasonlítás? Ha például egy időpontban a rendszerek állapotkombinációinak nincs közös eleme (arra az időpontra mást mondana az egyik, mást a másik), akkor azokat a rendszereket **ellentmondóknak** nevezzük.

(A tetszőleges absztrakt rendszerrel egyetlen rendszer sincs ellentmondásban.)

Lehetséges az is, hogy egy σ_A rendszer *minden* időpontban azonos vagy bővebb állapotkombinációt határoz meg, mint egy másik σ_B, s ekkor azt mondjuk, hogy a σ_A a σ_B **következménye** ($\sigma_A \Leftarrow \sigma_B$), ill. fordítva, σ_B a σ_A **feltétele** vagy **példája** ($\sigma_B \Rightarrow \sigma_A$) lesz.



40. ábra. Következmény és feltétel

(A tetszőleges absztrakt rendszer minden rendszer következménye, illetve annak minden rendszer a feltétele. Két rendszer pontosan akkor

egymás feltétele és következménye is ($\sigma_A \Leftarrow \sigma_B$ és $\sigma_A \Rightarrow \sigma_B$), ha azok azonosnak tekintettek ($\sigma_A \Leftrightarrow \sigma_B$). Minden rendszer egyben önmaga feltétele és következménye is. Minden rendszer a feltételének (példájának) a következménye.)

Ha a rendszer egy következménye nem a tetszőleges absztrakt rendszer, azt a rendszer **korlátozó megszorításának** vagy röviden **korlátozójának** nevezzük.

Rendszer következményét, azaz amikor bővítjük az időpontokban felvehető állapotkombinációkat (s ez lehet bizonyos időpontoktól történő eltekintés is, így a rendszerben több lesz a bizonytalanság) a kiindulási rendszer **hasonlítható absztrakciójának** is nevezzük. A rendszer feltétele más néven a **hasonlítható pontosítása**. Rendszer **példányának** pedig az olyan hasonlítható pontosítását (azaz olyan feltételét, példáját) nevezzük, amely az értelmezett időpontjaiban pontosan egy állapotot vesz fel.

Hasonlítható rendszerekkel műveletek is végezhetők, mégpedig az időpontokban felvett állapotkombinációikon végrehajtott halmaz-műveletekkel.

Nem ellentmondó hasonlítható rendszerek esetén (az időpontokban felvett állapotkombinációk metszetei alapján) képezhető azok **konjunkciója**. (A konjunkció minden kiindulási rendszer feltétele lesz.)

Hasonlóan, az unió műveletével képezhető a hasonlítható (akár ellentmondó) rendszerek **diszjunkciója** (mely eredménye minden kiindulási rendszer következménye lesz).

A halmazműveletekkel **rendszer ellentéte** vagy **negáltja** is képezhető, amely eredménye a kiindulási rendszer (T) időpontjaiban $P'(t) = S \setminus P(t)$, azaz „nem $P(t)$” állapotban van.

(Egy tetszőleges absztrakt rendszer ellentéte önmaga. Rendszernek létezik vele közvetlenül hasonlítható ellentéte. Rendszer és közvetlenül hasonlítható ellentétének uniója a hasonlítható tetszőleges absztrakt rendszer.)

A hasonlítható rendszerek így összehasonlíthatók: egyik lehet a másik következménye vagy feltétele, és ekkor azokat **függőknek** nevezzük (jelölés: $\sigma_A \wr \sigma_B$), lehetnek ellentmondók, vagy a harmadik esetben azokat **függetleneknek** nevezzük.

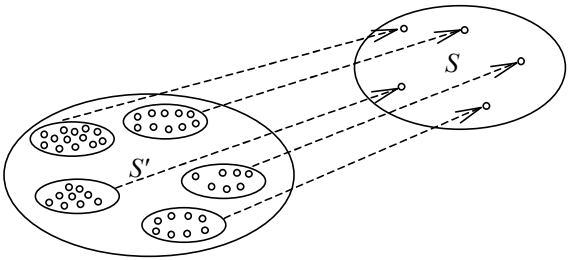
Absztrakció, pontosítás, típus

Az azonosnak tekintett rendszerek a *lényegében* azonos változást, az ekvivalens rendszerek pedig az azonos *jellegű* változás viszonyát írják le. A feltétel és a következmény a változás lényegének részletezettebb/vázlatosabb voltát adja meg. Ennek a mintájára a nagyobb, illetve kisebb változatosságú változások viszonya is definiálható, amit pontosításnak, illetve absztrakciónak fogunk nevezni. Kiindulási **rendszerek absztrakciójának** nevezünk egy rendszert, amely a kiindulási rendszerek (ekvivalens) átjelöléseinek a hasonlítható absztrakciója (következménye). Hasonlóan definiálható a **rendszerek pontosítása**. Rendszerek **nemdeterminisztikus absztrakciójának** nevezzük azt, amikor a kiindulási rendszerek (különböző) átjelölésein diszjunkció műveletet hajtunk végre.

(Két rendszer pontosan akkor ekvivalens, ha mindkét rendszer egymás általánosítása és pontosítása is egyszerre. Bármely rendszerből egy absztrakciót követő pontosítással tetszőleges rendszer képezhető.)

Bizony rendszerek állapotait (de nem állapotkombinációit) soroljuk osztályokba (azaz csoportosítsuk diszjunkt részhalmazokba), és az átjelölés során azokhoz rendeljünk egy-egy állapotot; valamint, a kiindulási rendszerek bizonyos időpontjaitól tekintsünk el! Ekkor egy speciális absztrakciót kapunk, amit **típusabsztrakciónak** nevezünk. A típusabsztrakció eredménye a **típus**, **minta** vagy **séma**. (Például, ha egy hőmérséklet-változást csak kiválasztott időpontokban tekintünk, illetve, ha az értékeket kerekítjük, típusabsztrakciót kapunk.)

(Típusabsztrakció a kiindulási rendszer következményéhez az eredményrendszer következményét rendeli.)



41. ábra. A típusabsztrakció a kiindulási állapotokat osztályokba csoportosítja

Ha egy kompozitból „elhagyjuk” az egyik komponenst (minden időpontban tetszőleges állapotú komponensre cseréljük), akkor a kiindulási rendszer következményét kapjuk eredményül. Ekkor „elvonatkoztattunk” a kiemelt résztől, „töröltük”, „figyelmen kívül hagytuk” azt a részt – ezt **egyszerű absztrakciónak** nevezzük. Az ezzel ellentétes átalakítás a **kiterjesztés**, vagy **egyszerű pontosítás**, amikor a rendszert újabb komponenssel, esetleg komponensekkel egészítjük ki.

(Bármely pontosítás eredményével ekvivalens rendszer megadható kiterjesztéssel.)

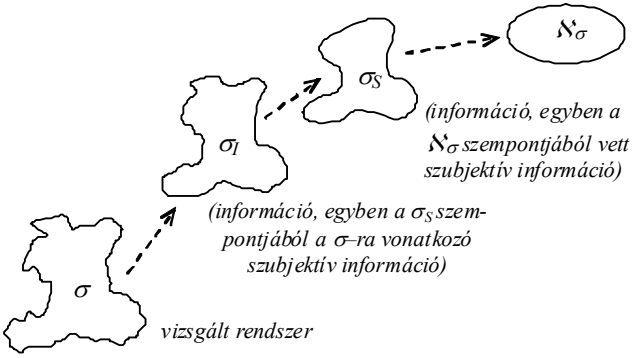
II.

Információ és ismeretszerzés

Információ

Rendszer minden *korlátozója* (azaz olyan következménye, amely nem tetszőleges absztrakt rendszer) a rendszer megközelítése abban az értelemben, hogy bizonyos pontossággal meghatározza az egyes időpillanatokban felvehető lehetséges állapotokat (azaz állapot-kombinációkat). Ezért, a **rendszerre vonatkozó információt** a rendszer korlátozójaként definiáljuk.

Ez a megfogalmazás nem kerül ellentétbe az információ hagyományos megközelítéseivel sem, ahol az információ az újdonsággal, a váratlansággal kapcsolatos. Egy információt egy másik következmény szempontjából vett **újdonságnak**, vagy **(szubjektív) információnak** nevezünk, ha nem annak a következménynek a következménye. (Az információ így a tetszőleges absztrakt rendszer szempontjából vett szubjektív információ, azaz egy „abszolút szempontból" vett újdonság. A rendszerre vonatkozó információ a rendszer következménye, azaz az információ feltétele a kiindulási rendszer.)



42. ábra: Információ és szubjektív információ.

Ábrázolás

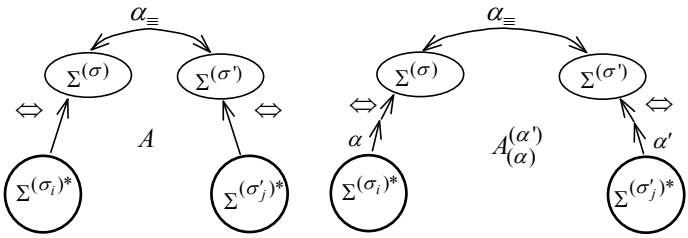
Azt már meg tudjuk fogalmazni, hogy mit nevezünk információnak, és hogy az mire vonatkozik (s ezt rendszernek nevezzük). Hogyan definiálhatjuk a rendszerek (és így egyben az információk) *ábrázolását*, amely az ábrázolt *rendszereket* és az *ábrázolásokat* kapcsolja össze?

Korábbi definíciónk alapján *közvetlen átjelölésnek* nevezzük a közvetlenül (!) hasonlítható rendszerek két halmaza közötti $\alpha_{\equiv} : \Sigma^{(\sigma)} \to \Sigma^{(\sigma')}$ függvényt, amely egy kiindulási rendszerhez a vele ekvivalens-rendszert rendeli. Ha a kiindulási rendszereknek, ill. külön, az eredményrendszereknek is megengedjük a hasonlíthatóvá alakítását, a közvetlen átjelölés kiterjeszthető (közvetett) **átjelöléssé** ($A : \Sigma^{(\sigma_{i\in I})^*} \to \Sigma^{(\sigma_{j\in J})^*}$), ami már hasonlítható (és nem csak közvetlenül hasonlítható) rendszereket kapcsol össze.

Még az *átjelölés* is feleslegesen „erős" eszköz, ugyanis olyan időpontokra, és állapotkombinációkra is meg kell adni az átjelölést, amelyek esetleg nem is jelennek meg azon $\sigma_{i\in I}$, ill. a $\sigma_{j\in J}$ rendszerekben, amelyek között meg szeretnénk teremteni a kapcsolatot. Többnyire nem törekszünk pontos átjelölésre, megelégszünk csak egy adott pontossággal.

Azt a függvényt, amely hasonlítható rendszerekhez azok egy-egy (hasonlítható) következményét rendeli, **absztrakciós módszernek** nevezzük ($\alpha : \Sigma^{(\sigma)^*} \to \Sigma^{(\sigma)^*}, \alpha(\sigma') \Leftarrow \sigma' \in \Sigma^{(\sigma)^*}$). Hasonlóan definálható a **pontosító módszer** is.

(Az absztrakciós módszer egy érdekes tulajdonsága, hogy hasonlítható rendszereken alkalmazva bővül a hasonlíthatóság köre, azaz $\Sigma^{(\sigma_{i\in I})^*} \subseteq \Sigma^{(\alpha(\sigma_{i\in I}))^*} \subseteq \Sigma^{(\aleph)^*}$.)



43. ábra: Átjelölés és absztrakt átjelölés.

Így már definiálhatjuk az **absztrakt átjelölést**, amely az együttese lesz a $\Sigma^{(\sigma_{i\in I})^*}$ hasonlítható rendszerek α absztrakciós módszerének, a $\Sigma^{(\sigma'_{j\in J})^*}$ hasonlítható rendszerek α' absztrakciós módszerének, valamint az $A:\Sigma^{(\alpha(\sigma_{i\in I}))^*}\to\Sigma^{(\alpha'(\sigma_{j\in J}))^*}$ átjelölésnek (jelölés: $A^{(\alpha')}_{(\alpha)}:\Sigma^{(\sigma_i)^*}\to\Sigma^{(\sigma'_j)^*}$).

Egy $A_{(\alpha)}$ absztrakt átjelölés (ha α' triviális, azaz minden rendszerhez azzal azonosnak tekintett rendszert rendel) a rendszereket csak egy adott absztrakciós szintig képes megközelíteni. (Például, egy autó digitális sebességmérője a sebesség tartományaihoz rendel egy-egy számértéket, így a sebességet csak a megfelelő absztrakciós szintig (tartományig) képes megközelíteni.)

Egy $A^{(\alpha')}$ (például $A_{(\alpha)}$ inverze) esetén a képek a kiindulási rendszerek absztrakt átjelöléseinél pontosabb rendszereket is meg tudnának közelíteni, de ekkor azok által már nem értelmezett időpontokra vagy állapot-kombinációkra kellene hivatkozniuk. (Például, egy digitális érték analóg módon történő kijelzése.)

Ha az α és α' is triviális, akkor az A egy (közvetett) átjelölés.

(Absztrakt átjelölés (az $\alpha_{\equiv}$ kiterjesztésével) kiterjeszthető átjelöléssé. Az átjelölés pedig absztrakt átjelöléssé szűkíthető. Tehát egy absztrakt átjelölés „hátterében" átjelölések állnak, egy átjelölés pedig absztrakt átjelöléseket „tartalmaz".)

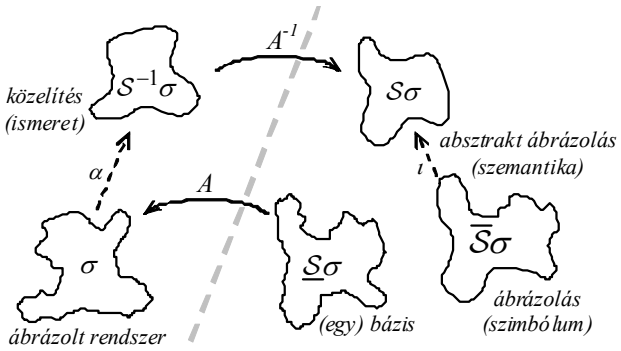
Az $A_{(\alpha)}:\Sigma^{(\sigma_{i\in I})^*}\to\Sigma^{(\sigma_{j\in J})}$ absztrakt átjelölések a kiindulási rendszerek vázlatos (áttekintő) „átjelölését", ábrázolását állítják elő, ezért azokat **absztrakt ábrázolási eljárásnak** nevezzük.

Ha választunk egy absztrakt ábrázolási eljárást, akkor egy kiindulási **rendszer absztrakt ábrázolásának** tetszőleges eredményrendszert fogunk nevezni. Azaz bármit bármi ábrázolhat, de ábrázolást csak egy rögzített absztrakt ábrázolási eljárásra és egy kiindulási rendszerre vonatkozóan vizsgálhatunk.

Az $A_{(\alpha)}$ absztrakt ábrázolási eljárás (korábbi megjegyzésünk alapján) kiterjeszthető többféle olyan A átjelöléssé, ami lehetővé teszi az invertálást, így egy absztrakt ábrázolásnak meg tudjuk keresni a kiindulási rendszerekkel hasonlítható képét. Egy kiindulási σ rendszer absztrakt ábrázolása legyen

most egy tetszőleges σ'. A kiindulási rendszer $A(\sigma) \in \Sigma^{(\sigma_j)^*}$ képét (azaz egy kiterjesztéssel kapott ábrázolást) az absztrakt ábrázolás (egy) **bázisának**, az absztrakt ábrázolásra alkalmazott $A^{-1}(\sigma')$ eredményt (az ábrázolás „visszajelölt" inverz képét) pedig a kiindulási rendszer **közelítésének** vagy **ismeretnek** nevezzük.

Rendszer, illetve információ ábrázolását így a következőképpen definiálhatjuk: Egy $A_{(\alpha)} : \Sigma^{(\sigma_{i\in I})^*} \to \Sigma^{(\sigma_{j\in J})}$ absztrakt ábrázolási eljáráshoz megadunk egy ($\iota : \Sigma^{(\sigma_j)^*} \to \Sigma^{(\sigma_j)}$) **értelmezésnek** nevezett módszert. Az értelmezés egy **ábrázolásnak**, **reprezentációnak**, **jelnek** nevezett rendszerhez egy absztrakt ábrázolást – a jel **jelentését** vagy **szemantikáját** – rendeli.



44. ábra: Absztrakt ábrázolás és ábrázolás.

Ha az értelmezés invertálható, akkor egy jelhez pontosan egy szemantika, illetve egy szemantikához pontosan egy jel tartozik. Ellenkező esetben egy szemantika ősképe a jelek egy ($\iota^{-1}(\sigma') \subseteq \Sigma^{(\sigma_j)^*}$) halmaza, azaz a lehetséges jelek (mint rendszerek) absztrakciója, amit a szemantika **absztrakt jelének** nevezünk. Az absztrakt jel az azonos szemantikájú jelek egy csoportját fogja meghatározni. (Pl., a különböző módon leírt 1 számjegyek együttesen absztrakt jelet alkotnak.)

Egy $A_{(\alpha)}$ absztrakt ábrázolási eljáráshoz és ι értelmezési módszerhez választhatunk egy $\overline{\beta}$ pontosító módszert, amely az absztrakt jelek (azaz a

lehetséges változatok) közül kiválaszt pontosan egy konkrét jelet. Ekkor a $\overline{\beta}(\iota^{-1}(A_{(\alpha)}\sigma))$ összetett függvényt **ábrázolási eljárásnak** nevezzük. Az ábrázolási eljárás meghatározza az ábrázolandó kiindulási rendszer absztrakt jelét, valamint azok közül kiválaszt egy konkrét jelet, azaz egy tényleges ábrázolást. Egy $A_{(\alpha)}$ és egy ι értelmezés által megadott (de tetszőleges $\overline{\beta}$ pontosító módszerű) ábrázolási eljárást $A_{(\alpha)}^{(\iota)}$ módon jelölünk, ami így ábrázolási eljárásokat csak az absztrakt jel erejéig határoz meg.

Az $A_{(\alpha)}^{(\alpha')}$ absztrakt átjelölés egyben egy ábrázolás is, amit **egyszerű ábrázolásnak** nevezünk (ekkor az értelmezés egy α' absztrakciós módszer, amely egy ábrázoláshoz szemantikaként annak hasonlítható absztrakcióját rendeli).

Szempont, pontosság és eltérés

Bármi tehát bármit ábrázolhat. Hogyan tudjuk megmondani, hogy melyik ábrázolás a pontosabb? Adott ábrázolási eljárás esetén a rendszer **egzakt** vagy **pontos ábrázolásának** nevezzük azt az ábrázolást, amely szemantikája azonosnak tekinthető az ábrázolás (bármely) bázisával (ekkor a bázisok azonosnak tekinthetők).

(Az átjelölés (azaz a rendszer ekvivalens képe) egzakt ábrázolás.)

Közelítő ábrázolásnak nevezzük azt az ábrázolást, mely szemantikája bármely bázis (nem triviális) következménye. A nem egzakt és nem közelítő ábrázolás a **hibás ábrázolás**. Hasonlóan definiálható az **egzakt**, **közelítő** és **hibás ismeret** (más néven: közelítés) is, a kiindulási rendszer és az ismeret összevetésével.

Az ismeretek, ill. ábrázolások finomabban is minősíthetők, ha nem csak azok viszonyát, hanem eltérésüket is meg tudjuk határozni. Hasonlítható rendszerek egymástól való eltérése azonban más *szempont* esetén más és más lehet. A szempontot egy olyan függvényként adjuk meg, amely minden kiindulási rendszerhez annak egyre absztraktabb és absztraktabb következményét rendeli. A definíció a következő:

Hasonlítható rendszerek $\prec$ (ejtsd: „v") **szempontját** vagy **nézőpontját** egy (adott, $i0$-al jelölt minimumú és $i\infty$-el jelölt maximumú) rendezett

értékekhez rendelt olyan absztrakciós módszerekként adjuk meg, amely minimum esetén minden rendszerre vele azonosnak tekintett rendszert ($\prec_{(i0)}: \sigma_i \mapsto \sigma_i^{\Leftrightarrow}$) képez, maximum esetén az absztrakt rendszert képzi ($\prec_{(i\infty)}: \sigma_i \mapsto \aleph$), egy kisebbhez viszonyított nagyobb értékhez pedig következményt ($a \le b$ esetén $\prec_{(a)}(\sigma) \Rightarrow \prec_{(b)}(\sigma)$, azaz egyre absztraktabb rendszert) határoz meg.

(Adott számban (a 0-t is beleértve), egymás után végrehajtott absztrakciós módszerek sorozata szempont, ha határértéke a tetszőleges absztrakt rendszert eredményező absztrakció.)

A szempont fogalmával már definiálható a **rendszerek eltérése** (jelölés: $\Delta^{(\prec)}_{j\in J}\sigma_j$) úgy, hogy a rendszereken végrehajtjuk az absztrakciós módszereket és megkeressük a szempontnak azt a legkisebb (index-) értékét, amelyhez tartozó absztrakció már azonosnak tekintett rendszereket fog eredményezni. A rendszerek eltérése tehát egy index-érték (azaz érték) lesz, amely azonos szempont esetén más eltéréssel is összehasonlítható. Rendszerek (például egy rendszer és a rá vonatkozó ismeret) **pontosságának** pedig az eltérésükhöz tartozó rendszert nevezzük (jelölés: $|\sigma_a, \sigma_b, ...|_{\prec}$). Ez tehát a rendszereknek a szempont által kapott olyan absztrakt képe, amely már megegyezik.

(A pontosság a kiindulási rendszerek következménye. A pontosság maximális eltérés esetén a tetszőleges absztrakt rendszer. Azonosnak tekinthető rendszerek eltérése minimális (azaz *i*0), pontosságuk pedig a rendszerekkel azonosnak tekinthető.)

(Hasonlítható rendszereknek létezik a pontossága bármely szempont esetén, és az a $\aleph$ tetszőleges absztrakt rendszer példája.)

Ha a szempont értelmezési tartományaként a kiterjesztett nem-negatív (nullától végtelenig terjedő) valós számok halmazát, vagy annak egy megfelelő részhalmazát választjuk, akkor a hasonlítható rendszerek körében a ***szempont szerinti eltérés egy távolságot ad meg*** (!), így beszélhetünk a **rendszerhez** (a **szempont szerint**) **jobban vagy kevésbé hasonlító rendszerekről**. Valamint teljesülnek a távolság jellemzői is: (1) Két rendszer távolsága nem negatív és pontosan akkor 0, ha a két rendszer azonosnak tekinthető. (2) σ_a és σ_b távolsága azonos σ_b és σ_a távolságával. (3) σ_a

és σ_b, valamint σ_b és σ_c távolságainak összege legfeljebb σ_a és σ_c távolsága.

(Ábrázolás szemantikájának következménye az ábrázolás pontossága. Egzakt ábrázolás pontosabb bármely nem egzakt ábrázolásnál bármely szempont esetén. Ismeret pontossága – mint ismeret – egy nem hibás ismeret bármely szempont esetén.)

(Függő és nem hibás ismeretek esetén, ha az egyik ismeret a többinél pontosabb egy adott szempont szerint, akkor bármely szempont szerint (!) legalább olyan pontos, mint a többi ismeret.)

(Nem hibás ismeretek nem lehetnek ellentmondó rendszerek.)

(***Független*** (azaz nem hibás és nem függő) ***ismeretek esetén létezik olyan szempont, melyben az egyik, és létezik olyan, amelyben egy másik ismeret a pontosabb***; *ekkor az ismeretek konjunkciója – mint ismeret – bármely szempont esetén legalább olyan pontos, mint a többi ismeret.*)

Így eszközeinkkel definiálhatjuk adott szempont esetén egy **rendszer ábrázolásának** és egyben az **ismeretnek az információtartalmát**, mint az ábrázoláshoz tartozó ismeret (közelítés) szempont szerinti pontosságát.

(Az ábrázolás, ill. az ismeret információtartalma, ha az nem a tetszőleges absztrakt rendszer, akkor a kiindulási rendszerre vonatkozó információ.)

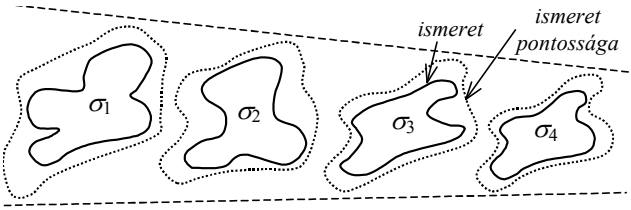
Ismeretszerzés

Vegyünk egy rendszert és annak olyan $\sigma_{i\in I}$ következményeit (ahol *I* egy *i0* minimumú rendezett indexhalmaz), ahol minden „későbbi” (nagyobb indexű) rendszer a hasonlítható pontosítása a „korábbiaknak”, azaz a rendszerek nem térnek el a kiindulási rendszertől. Ezt a sorozatot a rendszer **monoton közelítésének** nevezzük. Ha a kiindulási σ_{i0}-nál létezik legalább egy pontosabb a sorozatban, akkor a közelítést **kontrakciónak** nevezzük.

Adott σ rendszerre és egy $\prec$ szempontra vonatkozó **ismeretszerzést** vagy **tanulást** az ismeretként tekintett rendszerek olyan sorozataként definiáljuk, melyek (a szempont szerinti) pontosságának konjunkciói (amiket a **tanulás pontosságainak** nevezünk) kontrakciót alkotnak.

Ismeretszerzés esetén mindössze az ismeretek pontosságainak kell kontrakciót alkotniuk. Itt tehát a folyamatban lehetséges a közelítendő rendszertől eltérő (független vagy akár ellentmondó) ismeret is, mivel mindössze azok pontosságainak viszonyait vizsgáljuk. Például, a rendszerről egy függő ismeret előállítása is lehet tanulás, ha annak pontossága a kiindulási ismerettől kevésbé tér el.

Egy kontrakció éppen ezért egy, a szempontoktól független „ideális ismeretszerzés" (más szemszögből: az ismeretszerzések ideálizált változata), amely soha nem tér el a kiindulási rendszertől, hanem annak a többé-kevésbé pontosabb absztrakt képét állítja elő.



45. ábra. A tanulás a rendszer pontosabb körvonalait határozza meg.

(Hőmérsékleti értékek esetén például a kezdeti [20C°;24C°] tartományt követő [22C°;25C°] „ismeret" a tényleges [22C°;23C°] tartományra vonatkozó ismeretszerzés lesz. Ekkor a kezdeti pontosság [20C°;24C°], a következő mérés pontossága [22C°;25C°], melynek a korábbi ismeret pontosságával vett konjunkciója [22C°;24C°], így a pontosságok ismételt konjunkciói kontrakciót alkotnak.)

Mivel a rendszerek (rendezett halmazon értelmezett) sorozata is folyamat, így *az ismeretszerzés rendszerként vizsgálható*.

Példákon keresztül történő ismeretszerzés

Általában egyetlen lépésben nem tudjuk megismerni a vizsgált rendszert. Bonyolult rendszerek közelítését csak egyes eleminek, zártnak, „ismertnek" tekintett részletek összeépítésével, az elemek – részleges tudásunkkal és áttekintőképességünkkel megfogható – összekapcsolásával tudjuk megadni. „Ismert" kiindulási rendszerekből adott, „ismert" módon építkezünk, ezért az eredményül kapott rendszer is „ismert" lesz. Így, ha az ismeretünk a

közelíteni kívánt rendszerrel azonosnak tekinthető módon változik, akkor a keresett rendszert megismertük. *Az ismeretszerzés* így valójában *egy ábrázolási eljárást is jelent*.

Az ismeretszerzéshez egyrészt szükséges az ismeret tesztelése, hogy az mennyiben egyezik meg, illetve mely részleteiben tér el a közelíteni kívánt rendszertől (*összehasonlítás*), másrészt szükséges az ismeretnek az eltérés alapján történő *módosítása*. Egy σ rendszer megismeréseként tekintsük most a következő módszert: Vegyünk egy tetszőleges ismeretet (például a tetszőleges absztrakt rendszert ($\aleph$))! Ha az ismeret túlságosan *absztrakt* (a közelítendő rendszer következménye), akkor hajtsunk végre azon egy hasonlítható pontosítást! Ha az ismeret túlságosan pontos (a közelítendő rendszernek mindössze a *példája*, azaz nem tartalmazza az összes lehetséges példát), akkor hajtsunk végre egy hasonlítható absztrakciót, célszerűen olyat, amely a hiányzó példákat is tartalmazza! Ha a rendszerek nem függők, akkor hajtsunk végre egy olyan hasonlítható absztrakciót, amely eredménye az ismeret mellett a közelítendő rendszer hasonlítható absztrakciója is lesz! Ha az ismeret egzakt, azaz a közelítendő rendszerrel azonosnak tekinthető, akkor a rendszert megismertük.

Az ismeret módosításának egy lépéséhez tehát elegendő egy hasonlítható absztrakció vagy egy hasonlítható pontosítás. A hasonlítható absztrakció megadható egy (megfelelő) kiegészítő rendszer diszjunkciójával. A hasonlítható pontosítás pedig megadható egy (megfelelő) kiegészítő rendszer konjunkciójával. Ha egy példa a rendszerben nem, de az ismeretben szerepel, akkor pedig a példa ellentétének konjunkciójával pontosíthatunk a rendszeren. Az ismeret módosításához szükséges eszközeinket tehát csökkenthetjük a jelenlegi ismeret és egy kiegészítés konjunkciójára és diszjunkciójára.

A közelítésnek a még hiányzó, illetve a felesleges példák keresésén alapuló ismételt módosításaival pontosabb ismereteket állíthatunk elő, így ez egy *ismeretszerzés*, egy tanulási folyamat, amit **példákon keresztül történő ismeretszerzésnek** nevezzük.

Analízis és szintézis

Megközelítésünkben az *analízis* fogalmat olyan értelemben használjuk, hogy egy nagyobb egységet adott kiindulási elemek adott módú összeállításaként

határozunk meg. Ehhez definiálnunk kell két fő fogalmat, azt, hogy miből építkezünk (s ezeket a rendszer fogalmainak nevezzük), és hogy milyen módon (ezt pedig módszernek fogjuk nevezni).

Ha egy rendszert a saját idejében és helyein (egy átjelöléssel) más helyre és időbe „tolunk át", azt **közvetlen pozícionálásnak** nevezzük. Ha más rendszer idejébe és helyeire helyezzük át, akkor ezt az átjelölést **pozícionálásnak** nevezzük. **Rendszer tényleges fogalmát** a korlátozásának következményeként definiáljuk, annak valamely közvetlen pozícionálása pedig a **rendszer lehetséges fogalma**.

Rendszer fogalmát úgy definiáljuk, hogy az egy tényleges fogalmának a megfelelő pozícionálása, mely pozícionálást a **fogalom alkalmazásának** nevezzük („abban a rendszerben hol és mikor van?").

(A tényleges fogalom egyben a rendszer lehetséges fogalma. Rendszer következménye és példája, valamint kompozit része is a rendszer tényleges fogalma. Rendszer tényleges és lehetséges fogalmai egyben a rendszer fogalmai is. *Példánynak* (így valószínűleg a valóságnak is) *a lehetséges fogalmai a tényleges fogalmai, azaz a rendszer következményei.*)

A rendszer működésében más időpontban, illetve más helyeken megjelenő ismétlődések kiemelhetők a rendszer egy lehetséges fogalmává, mely közvetlen pozícionálásával megadhatjuk annak tényleges időpontjait és helyeit.

Rendszer egy **fogalmának** a **tényleges tartományának** nevezzük a rendszer azon tényleges fogalmainak diszjunkcióit, melyek megfelelő pozícionálásaival a fogalommal azonosnak tekinthető rendszert kapunk („ott valóban manifesztálódik a fogalom"). Egy **fogalom lehetséges tartománya** a rendszer azon lehetséges fogalmainak diszjunkciói, melyek megfelelő pozícionálásaival a fogalommal azonosnak tekinthető rendszert kapunk („a fogalom ott így jelenhetne meg").

(Rendszer adott fogalmának lehetséges tartománya a tényleges tartományának következménye.)

A rendszer „tényleges fogalma" általában a rendszerre vonatkozó információnál bővebb fogalom. Egy (nem példány) rendszer egy nem triviális példája tényleges fogalom, de nem információ.

(Példány tényleges fogalmai legfeljebb a példány hasonlítható absztrakciói (mivel a példány nem korlátozható tovább), azaz a tetszőlegesség (א) kivételével mind a példányra vonatkozó információk.)

A fogalmakból hogyan építhetünk nagyobb rendszereket?

Hasonlítható rendszereknek elkészíthetjük adott indexhalmaz alapján a szorzatát (mely indexeket **szerepeknek** nevezünk); a szorzatok egy részhalmaza a **rendszerek közötti reláció** lesz (pl.: (férj, feleség, gyerek) szerepek esetén egy szorzat-elem: (férj: *James*, feleség: *Jane*, gyerek: *Jimmy*, gyerek: *Nancy*); a hasonló szorzatok relációját – az összes ilyen viszonyú kapcsolatot – pedig családnak nevezhetjük el). A szorzatok egy részhalmazához hasonlítható rendszert rendelő függvényt **hasonlítható rendszerek módszerének** nevezzük.

(Rendszerek konjunkciója, diszjunkciója, illetve rendszer negációja módszer. Módszer átírható relációvá.).

Így már van eszközünk az összetett megadásra: megvan a „miből" (fogalmakból) és a „hogyan" (módszerként). Így, ha egy rendszert módszerként adunk meg, azt **makrorendszerként való megadásnak**, a paraméter-szerepű fogalmakat pedig a makrorendszer **építő-** vagy **alkotóelemeinek** nevezzük. A módszer természetesen több, elemibb módszerből felépített összetett módszer is lehet.

Rendszer makrorendszerként történő megadását a kiindulási rendszer **analízisének** és egyben az alkotóelemek **szintézisének** nevezzük.

A lehetséges és tényleges rész

Megállapítottuk, hogy egy kiindulásként vett ismeret (mint közelítés) pontosítható (a példa „tesztjeinek" összehasonlítása alapján) az absztrakció, illetve a pontosítás irányában történő kiegészítéssel. A felvehető állapotkombinációk bővítésén, illetve szűkítésén alapuló makrorendszer-képzés pontosan a közelítések lépéseinek folyamatát követi. Ezért ez a módszer *adott rendszer egy közelítését az ismeretszerzés folyamatának rögzítésével együtt ad meg*. Ellentétes irányból tekintve: ***egy makrorendszer az ismeret mellett a tanulási folyamatot is megadja, azaz az ismertnek feltételezett fogalmak alapján „megismertet" az ismerettel.***

Adott ismeretszerzési folyamat kerülőútjai, illetve adott lépéseknek a későbbiekben absztrahált partikuláris eredményei átrendezhetők egy *„ideálisabb"* ismeretszerzési folyamattá. Ez a megadási mód ezért egy **absztrakt ismeretszerzési folyamatot** ír le, amelyben *kevesebb a kerülőút, és a nem*

absztrahált partikuláris ismeret, valamint, amelyben bizonyos lépések felcserélhetők.

A kiegészítő rendszerekkel történő pontosító vagy absztrakciós kiegészítések átírhatók a konjunkció, diszjunkció és negáció műveleteire, majd azok átalakíthatók (vonatkozó részekként, illetve „rétegenként"), bizonyos lépések *összevonhatók*, melyek eredményeként egy „ideálisabb", azaz kerülőutaktól és partikuláris eredményektől mentesebb ismeretszerzési folyamatot kapunk. Az összevonás egy lehetséges eredménye a konjunkciók diszjunkciójaként kapott **diszjunktív normálforma**, vagy a diszjunkciók konjunkciójaként kapott **konjunktív normálforma**.

Rendszer egy következményét **tényleges résznek** is nevezzük, egy rész feltételét (hasonlítható pontosítását) pedig **lehetséges résznek**.

(Rendszer által lehetséges és tényleges részként tartalmazott rendszerek a rendszer fogalmai. Rendszer komponense egyszerre a rendszer lehetséges és tényleges része.)

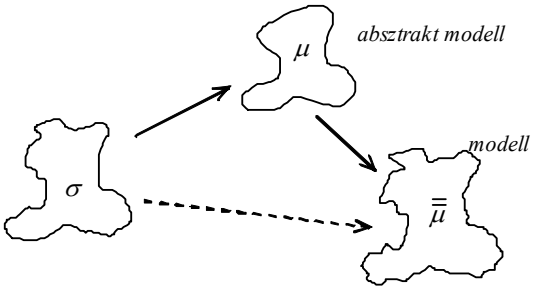
(Rendszerek diszjunkciója esetén a kiindulási rendszerek az eredmény lehetséges részei. Konjunkció kiindulási rendszerei az eredmény tényleges részei.)

III.

Racionalitás és analógia

Modell

Megközelítésünkben megkülönböztetjük a **rendszer absztrakt modelljét**, amit egyszerűen a rendszer típusabsztrakciójaként (függvény-jellegű kapcsolatként) definiálunk, és a **modelljét**, amely az absztrakt modell egy példánya, konkrét változata. (A modell tehát egy ábrázolás, mégpedig egyszerű ábrázolás, az absztrakt modell pedig absztrakt ábrázolás.)



46. ábra: Absztrakt modell és modell.

(Az absztrakt modell pontossága önmaga, mivel az egy közelítő vagy egzakt ábrázolás. Modell pontossága pedig a modell absztrakt modellje.)

(A tetszőleges absztrakt rendszer bármely rendszer absztrakt modellje.)

(Például, egy ember egy oszlopot – mellette elhaladva – megfog és ahhoz a karizma megfeszítésével egyre közelebb kerül, így a sebessége is nagyobb lesz. Ezzel megalkotta a bolygó mellett elhaladva felgyorsító mesterséges hold modelljét. A modell mögötti lényeg – a szemantika – az absztrakt modell, amely a fizika törvényeivel írja le a centripetális gyorsuláshoz szükségesnél nagyobb erővel körmozgásra kényszerített test kerületi sebességének növekedését.)

A típusabsztrakció egy „jól viselkedő” absztrakció, mivel a rendszer és típusa között egy függvény-jellegű kapcsolatot ír le, így a modellként történő ábrázolás is egy „jól viselkedő”, könnyen értelmezhető ábrázolás.

(Kompozit részeinek típusabsztrakcióján végrehajtott kompozíció a kiindulási rendszer absztrakt modellje is lesz: azaz kompozit modelljét a részeinek modelljeként is felépíthetünk.)

Kvantumrendszer, kvantumváltozás

Általános rendszer esetén az időpontok halmaza egy tetszőleges rendezett halmaz, így lehet például a valós számok folytonos halmaza is. Gyakran azonban elegendő, ha a változást csak adott időpontokban, adott „mérföldkövek” esetén vizsgáljuk, a közbülső időtartományban pedig tetszőlegesnek vesszük a változást. Az eredetileg folyamatos változás ekkor pillanatfelvételek sorozataként, ugrásszerű átmenetekként, az állapotok „kvantum-ugrásaiként” jelenik meg.

Egy rendezett halmazt **rendezett kvantumhalmaznak** nevezünk, ha bármely elemének meg tudjuk mondani a **rákövetkezőjét**, amennyiben az létezik. **Kvantumhalmaznak** nevezünk egy halmazt, ha azon definiálható olyan rendezés, mely esetén rendezett kvantumhalmazt kapunk.

(Egy rendezett halmaz pontosan akkor rendezett kvantumhalmaz, ha ahhoz kölcsönösen egyértelmű és szigorúan monoton módon hozzárendelhető az egész számok egy részhalmaza. Egy kvantumhalmaz legfeljebb megszámlálható számosságú.)

Rendszert **kvantumrendszernek** nevezünk, ha az értelmezett időpontok halmaza rendezett kvantumhalmaz. Ha a rendszert csak kvantum-időpontokban vizsgáljuk, azt **kvantált rendszernek** nevezzük. Rendszer kvantálásával kapott kép leegyszerűsíti mind a változás vizsgálatát, mind annak ábrázolását. (Kvantált rendszer a kiindulási rendszer absztrakt modellje, egyben kvantumrendszer.)

Ha egy rendszer esetén az értelmezett időpontok halmazának létezik minimuma, akkor azt **kiinduló-** vagy **startpontnak** nevezzük. (Startpontú kvantumrendszer változása ábrázolható az állapotok sorozataként: $s_0 \mapsto s_1 \mapsto s_2 \mapsto \ldots$ (s_0 utáni s_1 állapot, azt követő s_2 állapot, stb.), véges esetben: $s_0 \mapsto s_1 \mapsto s_2 \mapsto \ldots \mapsto s_n$.)

(Kvantumrendszer adott időpont utáni értelmezett változása megadható az állapotok sorozataként.)

Ha egy rendszert csak két időpontban vizsgálunk, az így kapott rendszer valójában a változás legkisebb egysége, s azt az eredeti rendszer **állapotváltásának** vagy **lépésének** nevezzük. A lépés tekinthető állapot-transzformációként is, amely a *t* értelmezett időpontban felvett *s* állapothoz a *t′*-ben felvett *s′* állapotot rendeli.

Ismeretszerzés mint rendszer

Az ismeretszerzés maga is folyamat, azaz rendszer, ahol a rendezett I halmazt tekintjük időpontoknak, az ismereteket pedig állapotoknak.

(Ismeretszerzés pontosságai ismeretszerzést alkotnak.)

Ha egy ismeretszerzés *i* „időpontjaihoz" állapotként az ismeretszerzés $\prec$ szempontja szerinti $p_i^{(\prec)}$ pontosságokat rendeljük (melyek az ismeretszerzés eredményeinek tekinthetők), akkor az így kapott rendszer absztrakt módon írja le az ismeretszerzési folyamatot, ezért azt **absztrakt ismeretszerzési folyamatnak** nevezzük.

(Az absztrakt ismeretszerzési folyamat egy kontrakció, illetve minden kontrakció egy-egy absztrakt ismeretszerzési folyamat. A kontrakció, mint „ideális" ismeretszerzés ezért alkalmas arra, hogy egy ismeretszerzést csak az eredményeinek tekinthető pontosságainak erejéig határozzon meg, azaz egy kontrakció több ismeretszerzés halmazát, egy *absztrakt ismeretszerzést* ad meg.)

A σ rendszer $\sigma_{i\in I}$ monoton közelítését a σ **szigorúan monoton közelítésének** nevezzük, ha egyre pontosabb rendszerekkel közelít, azaz ha $\forall i < j$ $(i, j \in I)$ esetén σ_i és σ_j nem azonosnak tekintett rendszerek.

(Egy szigorúan monoton közelítés kontrakció, ha az legalább kételemű.)

A σ rendszer esetén a (vele hasonlítható) $\sigma_{i\in N}$ (N a természetes számok halmaza, vagy annak $0\ldots n$ részhalmaza) ismeretek sorozatát a rendszer **lépésenkénti közelítésének** nevezzük.

Egy ismeretszerzési folyamat kvantálása azt jelenti, hogy a folyamatot csak bizonyos mérföldkövek esetén vizsgáljuk, a köztes állapotoktól pedig eltekintünk. *Kvantálással így egyszerűbbé válik az ismeretszerzések ábrázolása.*

(Kvantált ismeretszerzési folyamat ábrázolható sorozatként. *Kvantált absztrakt ismeretszerzési folyamat ábrázolható a pontosságok sorozatként.)*
(Kvantált ismeretszerzési folyamat egy lépésenkénti közelítés.)

Az ismeretszerzés kvantálásakor egy kiválasztott következő mérföldkő lehet például az újdonságot jelentő pontosságú új σ_i ismeret, azaz amely pontossága nem az eddigi $p_i^{(\prec)}$ pontosságok következménye.

(Kontrakció kvantálható legalább kételemű, szigorúan monoton közelítéssé, mely maga is egy „ideális" ismeretszerzési folyamat.)
A mérföldkövek megfelelő kiválasztásával elhagyhatók az újdonságot nem hozó köztes ismeretek, így ezen a módon szigorúan monoton közelítést kapunk. Így az eredmény egy „ideálisabb", ill. könnyebben ábrázolható ismeretszerzés lesz.

Lépésenkénti véges választású közelítés

Megközelítésünkben gyakran jelenik meg a *választás* mint eszköz, azaz egy halmaz adott részhalmazának a meghatározása. A rendszer definíciója maga is kiválasztásokon alapszik, mivel az egyes időpontokban a teljességet jelentő teljes állapothalmaz egy-egy részhalmazaként adjuk meg az időpontban lehetséges állapotok halmazát. A hasonlítható pontosítás ugyancsak kiválasztásokon alapszik, mivel szűkíteni kell az egyes időpontokban felvehető állapotkombinációkat, azaz ugyancsak egyes halmazok részhalmazát kell meghatározni. Az ismeretszerzésnek a kezdőpont ismereténél pontosabb rendszert kell meghatározni, ezért a kiválasztás az ismeretszerzés folyamatában is alapvető szerepet játszik. Választáson általában egy kvantált műveletet értünk (amely egy halmazhoz annak egy részhalmazát rendeli), de a „folytonosan lezajló választás" is értelmezhető.

A kiválasztás módszerét tovább korlátozhatjuk, ha egy lépésben csak *véges számú lehetőségből* történő választást engedélyezünk, amely pedig tovább bontható egyszerű igen/nem döntésekké. A lépésenkénti választások

módszere egy sajátos ábrázolási módot implikál, a véges szimbólum-halmazból képzett sorozattal történő ábrázolást, amelyet **leírásnak** fogunk nevezni.

Egy vizsgált *determinisztikus megadású* rendszer közelítését hajtsuk végre a következő módon! A rendszer lehetséges állapotait osztályozzuk néhány (a „néhány" jelzőt a következőkben a „véges számú" szinonimájaként használjuk) absztrakt állapotba, majd a rendszer idejéből kiválasztva egy időpontot, határozzuk meg, hogy ott a rendszer melyik osztálynak megfelelő absztrakt állapotban van. Ezután vagy a rendszer idejéből egy újabb időpontot kiválasztva az eddigi osztályozás alapján meghatározzuk a rendszer ott felvett absztrakt állapotosztályát, vagy egy már kiválasztott időpontban meghatározott állapotot pontosítsunk úgy, hogy az absztrakt állapotosztályt további néhány (azaz véges számú) alosztályra bontjuk. Egyetlen feltételünk, hogy az absztrakt állapotok osztály-hierarchiája a teljes rendszerre egységes legyen, azaz egy osztálynak csak egyetlen alosztályokra bontását engedélyezzük. A módszer minden egyes lépése után kapott rendszer egyre pontosabban közelíti meg a vizsgált rendszert. Minden rendszer értelmezett időpontjainak és állapotainak halmazai – mivel véges számú osztályt véges lépésben véges alosztályokra bontottunk – véges halmazok. A rendszerek így kapott sorozatát ezért a vizsgált **rendszer lépésenkénti véges választású közelítésének** nevezzük. Minden lépés eredménye az előző rendszerek típuspontosítása, egyben a vizsgált rendszer típusabsztrakciója, azaz absztrakt modellje. Egy ilyen közelítő rendszer állapothalmaza az osztály-hierarchia legalsó szintjén álló alosztályokat tartalmazza. Ha egy kiválasztott időpontban nem határoztuk meg ennyire pontosan az állapotot, akkor ott – nemdeterminisztikus megadási módon – felvehető állapotként az állapotosztály összes legalsó szintű alosztályát fel kell sorolni.

A σ rendszer $\sigma_{i\in I}$ kontrakcióját a σ **tetszőleges pontosságú közelítésének** nevezzük, ha $\forall i < j$ esetén σ_i és σ_j nem azonosnak tekintett rendszerek (σ_i a σ_j nem triviális feltétele) vagy σ_j és σ azonosnak tekintett rendszerek. A tetszőleges pontosságot abban az értelemben használjuk, hogy bármely, még nem egzakt ismeretnél minden „későbbi" ismeret (nem triviálisan) pontosabb közelítés lesz.

(A lépésenkénti véges választású közelítés egy tetszőleges pontosságú közelítés.)

(A lépésenkénti véges választású közelítés egy kontrakció, és így egyben egy ismeretszerzési folyamat.)

Szimbolikus modell

A választásoknak, és így egyben az ismeret ábrázolásának egy lehetséges módja, ha minden választáshoz felveszünk egy megfelelő számosságú (véges) szimbólumhalmazt, a lehetőségekhez (alosztályokhoz) hozzárendelünk egy-egy szimbólumot és a választást a megfelelő szimbólummal jelöljük. Az ismeret ekkor a választott elem és a szimbólumhalmaz – mint érték és típusa – elem-kettőseiből alkotott sorozattal reprezentálható. Az ábrázolás egy másik módszere, ha egy (véges) alapszimbólumhalmazból képzett véges sorozatokat (a választások indexként szolgáló „elnevezéseit", azonosítóit) rendelünk a típushierarchia egyes választási lehetőségeihez.

(Jegyezzük meg, hogy determinisztikus és általános megadású rendszerek esetén is igaz, hogy a rendszer értelmezett állapotkombinációinak számossága legfeljebb az értelmezett időpontok számossága lehet.)

Rendszert **véges vagy korlátos rendszernek** nevezzük, ha a rendszer ideje véges (s ekkor az értelmezett állapotkombinációk halmaza is véges).

(A lépésenkénti véges választású közelítés minden lépésének eredményeként a rendszer egy-egy véges absztrakt modellje áll elő.)

Rendszer véges vagy korlátos szimbolikus modelljének ($\mathcal{M}^{\Sigma}\sigma$) nevezzük azt a $\mu = (T^{\Sigma}, S^{\Sigma}, P^{\mu})$ (véges) absztrakt modelljét, ahol T^{Σ} és S^{Σ} véges szimbólumhalmazok, valamint T^{Σ} rendezett.

(Rendszer véges szimbolikus modellje egyben a rendszer egy típusabsztrakciója.)

(Rendszer értelmezett időpontjainak véges részhalmazához szigorúan monoton módon, illetve állapothalmazához a természetes számok egy-egy véges részhalmazát rendelve a rendszer véges szimbolikus modelljét kapjuk a

természetes számok két részhalmazát egy-egy szimbólumhalmaznak tekintve.)

A véges absztrakt modellek bizonyos szempontból nem zártak, mivel egy *n* számú időponttal és *m* állapotosztállyal megadott absztrakt modell pontosítása már nem feltétlenül ábrázolható egy azonos számú időponttal és állapotosztállyal megadott véges modellel. A rendszerek pontosabb megközelítését lehetővé tevő *racionális szimbolikus modell* egy adott időponthoz nem csak egy véges szimbólumhalmaz elemét (egy hosszúságú „sorozatát"), hanem a szimbólumok véges sorozatát is rendelheti, illetve a rendszer időpontjait is szimbólumok véges és rendezett sorozataival jelöli el. Véges szimbólumhalmaz véges (de tetszőleges) hosszúságú sorozatai egy végtelen halmazt is alkothatnak, így a racionális rendszer végtelen sok időpillanaton lehet értelmezve és végtelen számú tényleges állapota is lehet, de minden időpont és az ott felvett állapot egy tetszőleges lépésenkénti véges választású közelítés véges számú lépése után pontosan meghatározható.

A véges és a racionális modellek között helyezkednek el a kvantált absztrakt modellek, ahol a rendszer minden időpontjához hozzárendelhető a szimbólumok véges sorozata, ugyanakkor megköveteljük, hogy minden, nem utolsó időpont esetén meghatározható legyen annak a „következője". Kvantumrendszert ezért **integrális rendszernek** is nevezzük, startpontú kvantumrendszert pedig **természetes rendszernek**.

(A véges rendszer egyben természetes rendszer is. A véges és a természetes rendszer egyben integrális rendszer is.)

(Rendszer kvantált absztrakt modelljei pontosan a rendszer integrális absztrakt modelljei, illetve, a rendszer kvantált modelljei pontosan a rendszer integrális modelljei.)

Rendszert **racionális rendszernek** nevezzük, ha a rendszer (értelmezett) időpontjai legfeljebb megszámlálhatók (ekkor az értelmezett állapotkombinációk halmaza is legfeljebb megszámlálható).

(A véges, természetes és integrális rendszer egyben racionális rendszer is.)

(Racionális absztrakt modell lépésenkénti véges választású közelítéssel történő pontosításának eredménye is racionális absztrakt modell.)

Rendszer $\mu = (T^{\Sigma^*}, S^{\Sigma^*}, P^{\mu})$ (racionális) absztrakt modelljét **racionális szimbolikus modellnek** nevezzük ($\mathcal{M}^{\Sigma^*}\sigma$), ha T^{Σ^*} és S^{Σ^*} a megfelelő T^{Σ} és S^{Σ} véges szimbólumhalmazokból képzett véges szimbólumsorozatok halmazai, valamint T^{Σ^*} rendezett. **Rendszer integrális szimbolikus modelljének** nevezzük a rendszer olyan racionális szimbolikus modellje, amely egyben kvantumrendszer. **Rendszer természetes szimbolikus modelljének** nevezzük a rendszer startpontú integrális szimbolikus modelljét (tehát ha az értelmezett időpillanatoknak létezik minimuma).

(Rendszer értelmezett időpontjainak megfelelő részhalmazához szigorúan monoton módon, illetve állapothalmazához a természetes számok részhalmazát rendelve a rendszer természetes szimbolikus modelljét kapjuk.)

(Rendszer értelmezett időpontjainak megfelelő részhalmazához szigorúan monoton módon, illetve állapothalmazához az egész számok részhalmazát rendelve a rendszer integrális szimbolikus modelljét kapjuk.)

(Rendszer értelmezett időpontjainak legfeljebb megszámlálható részhalmazához szigorúan monoton módon, illetve állapothalmazához a racionális számok részhalmazát rendelve a rendszer racionális szimbolikus modelljét kapjuk.)

A racionális absztrakt modellek bizonyos szempontból szintén nem zártak. Bár tetszőleges lépésenkénti véges pontosítás adott lépésének eredményét ábrázolni tudják, végtelen számú pontosítás eredményét, a minden határon túlnyúló közelítések határértékét azonban nem. A rendszerek pontosabb ábrázolását lehetővé tevő *valós szimbolikus modell* az időpontokhoz és az állapotokhoz a szimbólumok tetszőleges, akár végtelen sorozatát is rendelheti.

Rendszert ***valós rendszernek*** nevezzük, ha a rendszer ideje legfeljebb kontinuum számosságú (ekkor az értelmezett állapotkombinációk halmaza is legfeljebb kontinuum).

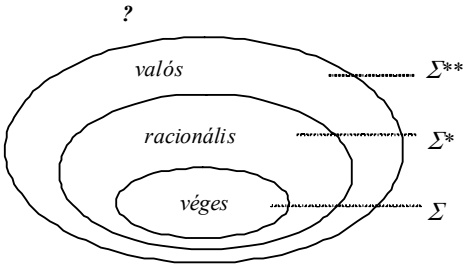
(Véges, természetes, integrális, vagy racionális rendszer egyben valós rendszer is.)

(A lépésenkénti véges választású közelítés módszerét tetszőleges, akár végtelen számú lépésben alkalmazva eredményül a rendszer valós absztrakt modelljét kapjuk.)

Rendszer $\mu = (T^{\Sigma^{**}}, S^{\Sigma^{**}}, P^{\mu})$ (racionális) absztrakt modelljét **valós szimbolikus modellnek** nevezzük ($\mathcal{M}^{\Sigma^{**}}\sigma$ vagy $\mathcal{M}^{\Sigma^{\infty}}\sigma$), ha $T^{\Sigma^{**}}$ és $S^{\Sigma^{**}}$ a megfelelő T^{Σ} és S^{Σ} véges szimbólumhalmazokból képzett tetszőleges (akár végtelen) hosszúságú szimbólumsorozatok halmazai, valamint $T^{\Sigma^{**}}$ rendezett.

(Rendszer értelmezett időpontjainak legfeljebb kontinuum számosságú részhalmazához szigorúan monoton módon, illetve állapothalmazához valós számok részhalmazát rendelve a rendszer valós szimbolikus modelljét kapjuk.)

A véges számú lehetőségből történő lépésenkénti választás egyszerű igen/nem döntésekre vezethető vissza. A diszkrét döntések megközelítési módszere alapvetően behatárolja a vizsgált rendszerről megszerezhető információk körét és a döntések száma alapján három, minőségileg különböző szintet határoz meg. Ha egy előre megadott értékkel korlátozzuk a döntések számát, akkor a vizsgált rendszer korlátos vagy véges absztrakt modelljét kapjuk. Tetszőleges, de véges számú döntés engedélyezése a racionális absztrakt modellt eredményezi, míg a döntések tetszőleges, akár végtelen száma esetén a valós modellhez jutunk.



47. ábra: Korlátos, racionális és valós absztrakt modellek.

A lépésenkénti véges választású közelítés alkalmas az értékek modellezésére. Az érték valójában egy index, amely a választási lehetőségek közül azonosítja a ténylegesen kiválasztottat. Az érték lehet hierarchikus (allehetőségek esetén, pl. „zöld” – „élénk zöld” – „nagyon élénk zöld”), így

pontossággal rendelkezik, illetve az érték lehet összetett (különböző szempontok esetén, pl. autóra: „gyors, zöld, kényelmes”).

Racionalitás és analógia

A *lépésenkénti véges számú lehetőségből történő választás* módszerének egyenes következménye a kiindulási rendszerről alkotott racionális, illetve ennek absztrahált határaként kapott valós kép. Létezik-e a valóson túli megközelítési mód, mellyel az információk bővebb köre szerezhető meg, illetve amellyel felgyorsíthatjuk a vizsgált rendszer közelítését? Minőségi változást csak úgy kaphatunk, ha lépésenként (gyakorlatilag potenciálisan-) *végtelen*, akár kontinuum számosságú lehetőségből történő választást is lehetővé teszünk. Hogyan képzelhető el egy ilyen, „analógiás” ismeretszerzés? Tekintsük a megismerendő rendszer minden egyes állapotát egy-egy összetett hullámként. A rezonancia, amely a hullámforma fő vonalait ábrázolni képes, végtelen számú lehetőségből választ. Az analógiás ismeretszerzés tehát elképzelhető egy „ráhangolódásként”, egy rezonancia-elven működő monoton közelítésként. A reprezentáció pedig elképzelhető úgy, mint ami az interpretáció során maga is összetett hullámformát gerjeszt (pl. egy vers hangzóváltásai, szavaihoz, szóösszetételeihez kapcsolódó formák a kiolvasás során egy összetett „hullámformát” generálnak). Az analógiás megközelítés egy gyorsabb és teljesebb körű ismeretszerzést tesz lehetővé, de az így szerzett bonyolult ismeretek teljeskörűen csak analógiás eszközökkel ellenőrizhetők, racionális eszközökkel nem.

Korlátai ellenére a racionális megközelítés több előnnyel is rendelkezik. Lassúsága ellenére a rendszer *tetszőleges pontosságú megközelítésére* is alkalmas, másrészt a közelítések könnyen dokumentálhatók, így ellenőrizhetők.

IV.

Hatás és kapcsolat

Absztrakt megadású rendszer

Eddigi rendszereink egy-egy, adott módon megtörténő változást írtak le, bár ez a változás lehetett bizonyos mértékig absztrakt, rendelkezhetett adott mértékű bizonytalansággal az időpontokban éppen felvett állapot tekintetében. A következőkben definiált „változó változás” különböző működéseket jelent.

A (közvetlenül hasonlítható) rendszerek $\sigma^{(\mathcal{A})} = (\mathcal{T}, S, P^{(\mathcal{A})})$ **absztrakt megadásakor** a $P^{(\mathcal{A})}$ minden $a \in \mathcal{A}$ argumentumhoz egy-egy $P^{(\mathcal{A})(a)} : T \to 2^S \setminus \{\varnothing\}$ (rövidítve: $P^{(a)}$) változást rendel, melyeket az absztrakt megadású rendszer **lefutásának** vagy **működésének** nevezünk. Az absztrakt megadási mód alkalmas a „változó változás”, azaz egy argumentumtól függően eltérő módozatú változás megadására.

Az argumentumok tetszőleges halmazt alkothatnak, általános esetben a rendszer világán „kívül is állhatnak”. Lehetnek azonban a rendszer bizonyos állapotkombinációi is, például az argumentum lehet egy kijelölt t_0 időpontban felvett s_0 állapotkombináció. Ebben az esetben egy $P^{(S_0)}$ formájú függvényt kapunk, amely minden t időponthoz megadja a t_0-ban felvett s_0 állapotkombinációjú rendszer $s_t^{(s_0)} = P^{(S_0)(s_0)}(t)$ aktuális állapotkombinációját (S_0 a t_0-ban felvehető állapotkombinációk halmazát jelöli). Az ilyen absztrakt megadású rendszert **átmeneti függvénynek** nevezzük.

(Bármely absztrakt megadású rendszer megadható átmeneti függvénnyel, egyszerűen úgy, hogy az argumentumot részként hozzávesszük a rendszerhez, mint egy paraméter-jellegű komponenst; ez úgy is tekinthető,

mintha a rendszert a paraméter-jellegű részbe mint környezetbe helyeztük volna.)

(Átmeneti függvénnyel megadott (absztrakt megadású) rendszer megadható egy egyoperandusú módszerként.)

Bizonyos startpontú kvantumrendszereket megadhatunk egy $P^{(\rightarrow)} : \overline{S} \rightarrow \overline{S}$ formájú függvénnyel úgy, hogy azt egy adott t_0-hoz tartozó $s_{t_0} \subseteq \overline{S}$ állapotkombinációval parametrizáljuk, ahol a $P^{(\rightarrow)} : \overline{S} \rightarrow \overline{S}$ formájú függvény minden *s* állapotkombinációból meghatározza a következő időpont s' kombinációját, amennyiben az időpont létezik. Az ilyen rendszereket **tiszta** (más néven: **reguláris**) **átmeneti függvénynek** nevezzük. Az *s* kiindulási állapotkombinációból a rendszer sorra az $s' = P^{(\rightarrow)}(s)$, $s'' = P^{(\rightarrow)}(s') = P^{(\rightarrow)}(P^{(\rightarrow)}(s))$, ... kombinációkba vált át.

(A tiszta átmeneti függvény egy átmeneti függvény)

Működések feltétele és korlátozója

Ha egy $\sigma^{(\mathcal{A})}$ absztrakt megadású rendszer argumentumainak veszzük egy bizonyos $\mathcal{C} \subseteq \mathcal{A}$ részét, akkor azt a $\sigma^{(c)}$ ($c \in \mathcal{C}$) **működések feltételének** vagy **feltételargumentumainak** nevezzük, a $\sigma^{(c)}$ ($c \in \mathcal{C}$) lefutásokat pedig a $\mathcal{C}$ **feltételű működéseknek**. (Mivel ez is absztrakt megadású rendszer, így használható a $\sigma^{(\mathcal{C} \subseteq \mathcal{A})}$, illetve rövidítve a $\sigma^{(\mathcal{C})}$ jelölés.)

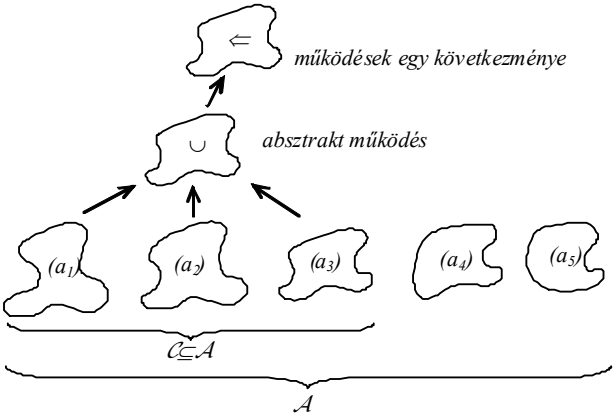
A $\sigma^{(\mathcal{C} \subseteq \mathcal{A})}$ működéseket összeolvasztva, azok $\bigcup_{c \in \mathcal{C}} \sigma^{(c)}$ unióját a $\mathcal{C}$ **feltételű absztrakt működésnek** (jelölés: $\bigcup \sigma^{(\mathcal{C})}$) vagy **feltételes absztrakt működésnek** nevezzük. Ez valójában a $\mathcal{C}$ feltételű lefutások lehetséges kerete, „medre”, azaz $\mathcal{C}$ argumentumok esetén a működések ezen az absztrakt rendszeren „belül” maradnak. Ha a feltétel a teljes argumentumhalmaz, akkor az összes lefutás $\bigcup \sigma^{(\mathcal{A})}$ unióját kapjuk, amit **feltétlen absztrakt működésnek** nevezünk.

Egy bizonyos σ' korlátozhatja egy $\sigma^{(\mathcal{C} \subseteq \mathcal{A})}$ minden egyes működését, amit ekkor a **működések** (feltételes vagy feltétlen) **korlátozójának**

nevezünk, vagy a **működések korlátozó feltételének**, mivel a σ' kiválogatja azokat a működéseket, amelyekre az teljesül. Ill. hasonlóan, beszélhetünk a **működések következményéről** (jelölés: $\sigma' \Leftarrow \sigma^{(\mathcal{C} \subseteq \mathcal{A})}$), ha a σ' mindegyik működés következménye. (Adott működések bármely következménye az adott működések korlátozója vagy a tetszőleges absztrakt rendszer.)

Adott (feltételű) működéseket tehát kiválogathatunk egyrészt az argumentumok egy részhalmazával (feltételargumentumok), mely működések együttes „medre" egy adott feltételű absztrakt működést határoz meg. Adott működéseket ugyancsak kiválogathatunk egy rendszer mint korlátozó feltétel, azaz a működések korlátozó feltételének a megadásával. A két fogalom szoros kapcsolatban van egymással.

(*Adott feltételű működéseknek a σ' pontosan akkor a következménye, ha az az adott feltételű absztrakt működés következménye*, azaz $(\sigma' \Leftarrow \sigma^{(C \subseteq A)}) \Leftrightarrow (\sigma' \Leftarrow \bigcup \sigma^{(C \subseteq A)})$).



48. ábra. Működések tartománya, ill. következmény feltételargumentumai

Általában véve, egy, a működésekkel hasonlítható rendszer (amit a működésekre vonatkozó ismeretnek neveztünk) kiválogat olyan működéseket, amelyekre az következményként (hasonlítható absztrakcióként) teljesül. Ezeknek a működéseknek az argumentum-részhalmaza (amire igaz, hogy

$\mathcal{C}_{\sigma'} = \{c \in \mathcal{A} \mid \sigma' \Leftarrow \sigma^{(c)}\}$) egy feltétel(argumentumhalmaz) lesz, ezért azt a **működésekre vonatkozó ismeret feltételének** nevezzük.

(A működésekre vonatkozó σ' ismeret feltétele üreshalmaz pontosan akkor, ha a σ' egyetlen működésnek sem következménye. A működésekre vonatkozó σ' ismeret feltétele a teljes $\mathcal{A}$ argumentumhalmaz pontosan akkor, ha a σ' a működések feltétlen következménye. A működésekkel hasonlítható tetszőleges absztrakt rendszer a működések feltétlen következménye.)

Entitás korlátozója

A következőkben **entitásnak** nevezzük egy kompozit valamely meghatározott részét (adott π_E helyeinek állapotváltozásait). Mivel egy kompozit egyben önmaga része is, ezért az entitásra vonatkozó fogalmak és állítások automatikusan a teljes rendszerre is alkalmazhatók.

Ha a $\sigma^{(\mathcal{C})}$ működéseket csak bizonyos kiválasztott $T' \subseteq \mathcal{T}$ időpontokban és csak egy E entitásra leszűkítve vizsgáljuk, azt az $E^{(\mathcal{C})}_{|T'} = \sigma^{(\mathcal{C})}_{|E|T'}$ módon írhatjuk le. Az E enitás $E^{(\mathcal{C})}_{|T'}$ **működéseinek tartományának** nevezzük az E entitás $\bigcup E^{(\mathcal{C})}_{|T'}$ absztrakt működését. *(Bővebb feltétel esetén a tartomány bővül*, azaz $\mathcal{C}' \subseteq \mathcal{C}$ esetén $\bigcup E^{(\mathcal{C}')}_{|T'} \subseteq \bigcup E^{(\mathcal{C})}_{|T'}$. Mivel bővebb argumentumhalmaz esetén az entitás változatosabb is lehet, ezért az absztrakt működése és így egyben a működéseinek a tartománya is bővülhet.)

Rendszer korlátozóját **funkcionális**- vagy más néven **értékkorlátozónak** nevezzük, ha az példány, tehát, ha az egyetlen értékre szűkíti a felvehető állapotokat. A nem értékkorlátozót (amely több felvehető értéket is megenged) **relációs**, **értéktartomány**- vagy más néven **tartománykorlátozónak** nevezzük.

Az $E^{(\mathcal{C})}$ működésű entitást a kiválasztott $T' \subseteq \mathcal{T}$ időpontokban vizsgálva **a változás függvénye** az időpontokban **korlátozza az entitás működését** (jelölés: $\rightarrow^{(\mathcal{C})}_{T'} E$), ha az $\bigcup E^{(\mathcal{C})}_{|T'}$ nem tetszőleges absztrakt rendszer (hanem az E korlátozója), ellenkező esetben **az entitás** az

időpontokban **korlátozatlan** (jelölés: $\not\rightarrow_{T'}^{(C)} E$). Az entitás korlátozója lehet érték- vagy tartománykorlátozó is.

Egy entitás bizonyos működéseinek korlátozása azt jelenti, hogy adott időpontokban az entitás által elfoglalt π_E helyeken a felvett állapotkombinációkat a tetszőlegességhez képest szűkíteni tudjuk.

(Entitás korlátozója a teljes rendszer adott működéseinek is a korlátozója, azaz a működésekre vonatkozó *információ*.)

Ha egy tulajdonság a rendszer összes értelmezett időpontjában fennáll, akkor azt **strukturális tulajdonságnak** nevezzük. Ellenkező esetben, ha találhatók olyan értelmezett időpillanatok, amikor a tulajdonság fennáll és olyanok is, amikor a tulajdonság sérül, akkor azt **időleges** (temporális) **tulajdonságnak** nevezzük. Strukturális tulajdonság jelölésekor elhagyható az összes értelmezett időpontok halmazának jelölése (pl. entitás strukturális korlátozója esetén: $\rightarrow^{(C)} E$).

Egy entitás működéseinek korlátozói, illetve következményei között megállapíthatjuk, hogy melyik az erősebb, illetve a gyengébb, természetesen egy adott szempont alapján. A $\prec$ szempont esetén, ha a működés egy $\prec_{(x)} (\sigma^{(c)})$ absztrakciójának a σ' már nem a következménye, de egy másik még igen, akkor σ' az **erősebb következmény**.

Két függő következmény esetén az „erősebb” a felvehető állapotoknak szűkebb részhalmazát határozza meg, míg a „gyengébbnek” nevezhető több állapotlehetőséget enged meg. (A tetszőleges absztrakt működés minden más következménynél gyengébb. *Működések következményének hasonlítható absztrakciója a működés gyengébb következménye.*)

(*Az absztrakció gyengíti a korlátozókat*, azaz adott működések következményénél a működések hasonlítható absztrakcióira gyengébb következmény adható meg.)

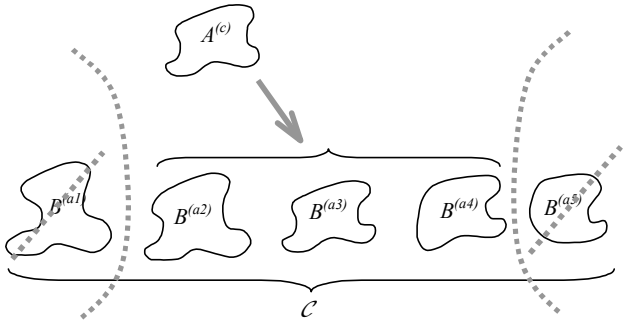
Hatás, függés, kapcsolat

A rendszer egyes részei, entitásai közötti hatást úgy fogalmazzuk meg, hogy a ható entitás korlátozza, „adott mederbe tereli” a függő entitás változását, így a ható entitás változása kikényszerítheti a függő entitás változását, ha az

még nincs a korlátozásnak megfelelő „mederben”. A hatás egy feltételes, *„ha – akkor”* korlátozás: *„ha”* a ható entitás adott működésű, *„akkor”* a függő entitásnak (más esetben lehetséges) bizonyos működései nem engedélyezettek.

A $\mathcal{C}$ feltételű működéseket vizsgálva egy A **entitás** (adott) $A^{(c)}$ **működésének hatását** bizonyos $T' \subseteq \mathcal{T}$ időpontokban egy B **entitásra** a következőképpen definiáljuk: Gyűjtsük ki a rendszer összes $\mathcal{C}$ feltételű működése közül azokat, amelyekben a kiválasztott időpontokban az A entitás az $A^{(c)}$ működéssel azonos módon változik, majd képezzük a kiválasztott működésekre a B ilyen feltételű változásainak az unióit (jelölés: $B_{(\leftarrow^{(\mathcal{C})}_{T'} A^{(c)})}$). A B entitás tehát csak ilyen lehet, ha az A entitás $A^{(c)}$ módon változik. Ezt összehasonlíthatjuk a B entitás $\mathcal{C}$ feltételű változásainak $\bigcup B^{(\mathcal{C})}_{|T'}$ tartományával, s ha az szűkebb (azaz $B_{(\leftarrow^{(\mathcal{C})}_{T'} A^{(c)})} \subset \bigcup B^{(\mathcal{C})}_{|T'}$), akkor azt mondjuk, hogy az A entitás $A^{(c)}$ (konkrét) működése a T' időpontokban **hat** a **függő** entitás $B^{(\mathcal{C})}$ működéseire (jelölés: $A^{(c)} \rightarrow^{(\mathcal{C})}_{T'} B$, ellenkező esetben: $A^{(c)} \not\rightarrow^{(\mathcal{C})}_{T'} B$). Több $A^{(\mathcal{C}')}$ működés hatását hasonlóan határozhatjuk meg.

(A feltétel bővülésével el is veszhet, de meg is jelenhet egy hatás.)



49. ábra. A hatás adott „mederbe tereli” a függő entitás változását.

Általános esetben a hatást/függést a következő módon fogalmazhatjuk meg: Ha a vizsgált időpontokban és vizsgált $\mathcal{C}$ feltétel esetén az A valamely (konkrét) $A^{(c)} \in A^{(\mathcal{C})}$ működése az időpontokban hat a $B^{(\mathcal{C})}$ működésekre,

akkor azt mondjuk, hogy A entitás **hat** a B entitás $B^{(C)}$ működéseire (jelölés: $A \rightarrow_{T'}^{(C)} B$), illetve a $B^{(C)}$ működések **függnek** az A entitástól (jelölés: $B \leftarrow_{T'}^{(C)} A$).

Adott időpontok és feltétel esetén a kölcsönös hatást ($A \leftarrow_{T'}^{(C)} B$ és $A \rightarrow_{T'}^{(C)} B$) **kölcsönhatásnak** nevezzük (jelölés: $A \leftrightarrow_{T'}^{(C)} B$). Ha a hatás irányától eltekintünk, azaz valamely entitás hat a másikra ($A \leftarrow_{T'}^{(C)} B$ vagy $A \rightarrow_{T'}^{(C)} B$), akkor azt mondjuk, a két entitás **kapcsolatban** van egymással (jelölés: $A \sim_{T'}^{(C)} B$). Ha a hatás az egyik irányban sem áll fenn, akkor az időpontokra vonatkozóan a két entitás **független** (jelölések: $A \not\sim_{T'}^{(C)} B$).

(Az entitások közötti hatás tranzitív, így egy entitás hatása több más entitásra is kiterjedhet. A kapcsolatok viszont nem feltétlenül tranzitívak. Például $A \rightarrow B$ és $A \rightarrow C$ esetén B és C független is lehet.)

(Változatlan működésű entitás nem hat entitásra és nem függ entitástól, így önmagára sem hat, illetve önmagától sem függ. Tetszőleges absztrakt rendszer egyetlen entitásra sem hat.)

Hatások erőssége

A hatás maga is korlátozásként jelenik meg, így korábbi definíciónk alapján beszélhetünk a *hatások mint korlátozók* adott szempont szerinti gyengeségéről, illetve erősségéről.

(*Az absztrakció gyengíti a hatásokat mint korlátozókat.* A működések hasonlítható absztrakciójára gyengébb következmények fogalmazhatók meg, így azokra gyengébb hatások mint korlátozók adhatók meg.)

A hatások között egy más jellegű gyengeség/erősség is definiálható. Absztraktabb működésre gyengébb korlátozók adhatók meg, ezért a rendszer egymást követő absztrakciói során „elvesznek” a hatások, így növekednek a függetlenségek. Adott szempont szerint **erősebbnek** nevezünk egy **hatást**, illetve **kapcsolatot**, ha az csak a szempont nagyobb mértékű absztrakciójánál tűnik el.

(*Hatás nem feltétlenül teljesül a működések következményére.* Egyrészt a ható entitás különböző működéseinek absztrakciói már megegyezhetnek, így egy működése azonossá válhat egy olyannal, mely már nem hat a függő entitásra. Másrészt a működések absztrakcióját tekintve a függő entitás működései is absztraktabbá válhatnak, így a hatás (mint korlátozó) is gyengül.)

(*Működések következménye esetén megjelenhet hatás*. Például, ha a poharat elengedem, és azt mondom „ess le”, akkor leesik; valójában a pohár egyébként is leesett volna, de egy absztrakció után úgy tűnik, mintha a „varázsszó” is szükséges lenne. A működések következményében megjelenő olyan hatást, amely az eredeti rendszerben nem állt fenn, a (**következményben megjelenő) látszólagos hatásnak** nevezzük.)

A **működések tartományabsztrakciójának** nevezzük az olyan hasonlítható absztrakciót, amelyeknél minden E entitás esetén a működések tartománya azonos marad ($\bigcup E^{(C)} = \bigcup E'^{(C)}$).

(Tartományabsztrakció esetén nem jelenhet meg új („látszólagos”) hatás. Tartományabsztrakció növeli a függetlenségeket.)

Enitás, környezete és külvilága

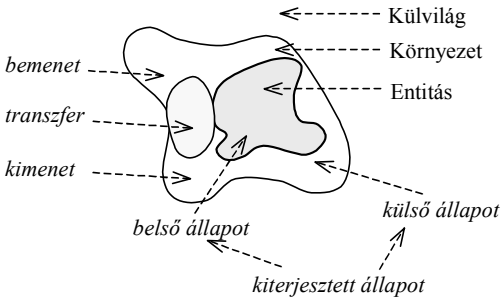
A rendszerből kiemelhetjük és vizsgálatunk tárgyául választhatjuk annak egy részletét, egy entitását. A kompozit így az *entitásra* és az azt körülvevő világra bontható a $\sigma = E \otimes E_{(\varnothing)}$ módon. Az entitást körülvevő világ azon részeit, amelyek hatnak az E entitásra az enitás **bemenetének** (jelölés: $E_{(\leftarrow)}$), amelyek pedig függnek az entitástól az entitás **kimenetének** (jelölés: $E_{(\rightarrow)}$) nevezzük. Entitás **transzfere** ($E_{(\leftrightarrow)}$) a rendszer azon részei, amelyek *egyszerre* hatnak és függnek is az entitástól (a bementében és kimenetében is szerepelnek). Az entitás **környezetét** ($E_{(\sim)}$) azon részek alkotják, melyek hatnak az entitásra *vagy* attól függnek (a bemenetében vagy a kimenetében szerepelnek). Az entitáson és környezetén kívüli részek az entitás **külvilága** ($E_{(\not\sim)}$), amely részek sem nem hatnak, sem nem függnek az entitástól, tehát nincsenek az entitással kapcsolatban.

(Entitás bemenete, kimenete, transzfere, környezete és külvilága egyaránt tekinthetők entitásoknak.)

Entitást és környezetét együttesen **kiterjesztett entitásnak** nevezzük.

(*Két entitás pontosan akkor független, ha egyik sincs benne a másik környezetében*, azaz egyik sem a környezetével együtt vett másik entitás részentitása.)

Rendszer egy működésének adott entitásra leszűkített változását az entitás **belső változásának**, adott időpontban felvett értékét az entitás **belső állapotának** nevezzük. Hasonlóan, a környezetre leszűkített változás, illetve állapot a **külső változás**, illetve a **külső állapot**. A környezetével együtt vett entitásra leszűkített változást, illetve állapotot pedig az entitás **kiterjesztett változásának**, illetve **kiterjesztett állapotának** nevezzük.



50. ábra. Entitás (komponens) és környezete.

(Ha megadjuk egy entitás kiterjesztett vagy belső változását, akkor a működések absztrakt modelljét kapjuk.)

(*Entitás meghatározza a működések kettős absztrakcióját*, ahol az első elvonatkoztat az entitás külvilágától, az annak eredményén végzett második absztrakció pedig az entitás környezetétől.)

(***Két egymásutáni tartományabsztrakció meghatározza a rendszer egy entitását*** a következő módon: Mindkét absztrakció során elveszhetnek részek közötti hatások. Válasszuk ki egy kompozit bizonyos részeit! A kiválasztott részekkel az első absztrakció után még mindig kapcsolatban lévő részeket együttesen a kiterjesztett entitásnak, a második absztrakció után a még mindig kapcsolatban lévő részeket együttesen pedig az entitásnak tekintjük. Entitás ilyen módszerű megkeresését az **entitás körvonalazásának** nevezzük.)

V.

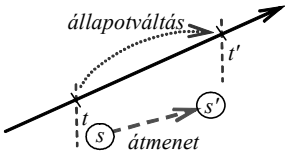
Időben lezajló változás

Esemény, átmenet és módosító

Ha egy rendszer változását egyetlen időtartam kezdő és záróponjtjára (a „bázis” és a „vizsgált” időpontban felvett állapotkombinációra) leszűkítjük, akkor – a korábbi definíciónk alapján – egy *állapotváltást* vagy *lépést* kapunk. Az állapotváltás lehet identikus is, amikor a bázis és a vizsgált időpont állapota megegyezik, azaz nem történt *tényleges* állapotváltás. A nem identikus állapotváltást **eseménynek** nevezzük, melynek van **ideje** (a vizsgált „későbbi” időpont), illetve – kompozit esetén – **helye** vagy **forrása**, amely a megváltozott entitás.

Az eseményeket és a működést a következőkben a *determinisztikus megadási* móddal vizsgáljuk.

Ha egy állapotváltásnak az időpontjaitól is eltekintünk, akkor a kezdő és záróállapot (s, s') kettősét kapjuk. Általánosan: egy S állapothalmaz elempárját ($(s, s') \in S \times S$) **átmenetnek** nevezzük (jelölés: $s \mapsto s'$), amelyet egy kétidőpontú (kvantum-) rendszernek tekintünk, ahol az s-et **kiindulási-** vagy **forrásállapotnak**, az s' -t **célállapotnak** nevezzük.



51. ábra. Állapotváltás és átmenet.

Absztrakt átmenetnek nevezünk egy nem üres állapothalmaz-kettőst (ami így megadható konkrét átmenetek halmazaként, azaz $S \times S$ nem üres részhalmazaként).

(Az állapotváltás megadható állapotátmenettel a megfelelő pozícionálás alkalmazásával, azaz a forráshelyek és időpontok megadásával).

Egy S állapothalmaz **módosítójának** vagy **transzformációjának** nevezzük az $m : S \to S$ leképezéseket.

(*Az állapot-transzformáció absztrakt állapotátmenetet határoz meg*, mivel egy forrásállapothoz – egyértelműen – egy célállapotot határoz meg.)

Transzformáció: átmenet és akció

Korábban megállapítottuk, hogy az absztrakt megadású rendszerek átírhatók az átmeneti függvények formájába, amelyek egy rögzített időponthoz viszonyítva adják meg a változást. Ezért igaz a következő:

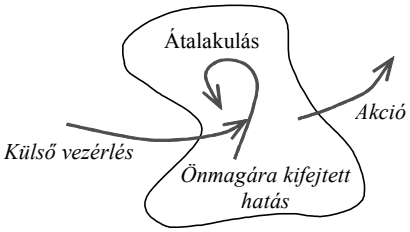
(*Átmeneti függvénnyel megadott startpontú kvantumrendszer átírható tiszta átmeneti függvény formájába*, azaz olyan rendszerré, amely átmeneti függvénye adott időpont esetén mindössze a rendszer állapotának ismeretében adja meg a következő időpontban (ha az létezik) felvett állapotot. Az ilyen rendszer változása átmenetekként jelenik meg, a „tiszta átmeneti függvény" pedig állapot-transzformációként, azaz $m : S \to S$ leképezésként, amely az adott időpontban felvett s állapot esetén megadja a következő időpontban felvett s' állapotot, az „állapot rákövetkezőjét".)

(Az állítás kiterjeszthető tetszőleges kvantumrendszerre, azaz: kvantumrendszer megadható tiszta átmeneti függvény formájában.)

(*Startpontú kvantumrendszer megadható a startpontban felvett* (determinisztikus esetben:) *állapottal és a megfelelő állapot-transzformációval.*)

Rendszer adott entitásának változását önmaga és környezete (bemenete) határozza meg, ugyanakkor a változása hatással lehet a környezetére (a kimenetre). Az entitás szempontjából vizsgálva az entitással kapcsolatos hatások alapján a teljes rendszer változása két komponensre bontható. Az egyik komponens a belső állapot változása, az **átalakulás** vagy belső átmenet, amelyet maga az entitás és annak környezete (pontosabban: a bemenete)

határoz meg. A másik komponens a rész hatása a környezetére (pontosabban: a kimenetre), amelyet a (rész) hatásának vagy **akciójának** nevezünk.



52. ábra. Külső vezérlés, átmenet és akció.

Rendszer adott entitásának adott időpillanatban felvett állapota így hatással lehet környezetének későbbi állapotára (akció), illetve a bemenettel együtt hatással lehet későbbi belső állapotára (átmenet).

Interakciók

Az entitásokra bontott rendszer, illetve az azon belüli hatások azonosítása a változások egy különleges megadását teszik lehetővé. Az *entitások rendszereként* történő megadás esetén a teljes rendszert entitásokra bontjuk fel, amelyek kapcsolatban lehetnek egymással (hathatnak egymásra), és ahol minden entitás a változását a bemenete és a saját állapota alapján határozza meg, illetve ezek alapján hathat a kimenetére.

Entitások-rendszereként a teljes rendszer változását egy adott formájú *meghatározottságként* adhatjuk meg, ahol a belső transzformációs szabályok alapján „automatikusan” történik a változás.

Kapcsolódó entitások esetén bizonyos hatások átvitelét azonnalinak tekintjük, pl. a kapcsoló felkapcsolására kigyullad egy izzó, a mérleg egyik oldalának lenyomására a másik oldal felemelkedik… Bizonyos kapcsolatok esetén azonban megjelenhet egyfajta közvetettség, mégpedig olyan módon, hogy a ható és/vagy a függő entitásnak kell kezdeményezni a hatás átvitelét. A hatás adott közvetettségen keresztül történő átvitele esetén így megkülönböztethetünk *aktív* és *passzív entitásokat*. A hatás átvitelét az aktív

entitás kezdeményezi, míg a passzív entitás csak válaszol az állapotára vonatkozó kérdésekre, illetve „elszenvedi” az egyes transzformációkat.

(Például, egy „aktív” pályázó „entitás” benyújt egy pályázatot egy hivatalnak, amivel adott folyamat lezajlását váltja ki. A pályázat eredményéről (mint annak egy absztrakt állapotáról) érdeklődhet; ebben az esetben ő az aktív fél. Másrészt a pályázat eredményéről kiértesíthetik; ebben az esetben a pályázó passzív. Siker esetén a megfelelő szerződés megkötése után hozzájut az adott forráshoz (passzív).)

Egy adott entitás tényleges állapotváltása (ami egy belső esemény), a külső entitások számára (külső-) eseményként jelenik meg, amely saját állapotuk megváltoztatására kényszerítheti őket. Összekapcsolódó entitások esetén így egy-egy változás egész eseménysorozatokat, egyben átmeneteket válthat ki.

Entitások rendszerében a következő eszközök szükségesek az *átmenetek megadásához*:

- az entitásnak
  - válaszolnia kell az absztrakt kiterjesztett állapotára vonatkozó kérdésekre, ill.
  - le kell tudnia kérdezni a környezetében (minimálisan a bemenetén) elhelyezkedő entitások absztrakt állapotát
- az entitásnak lehetővé kell tennie az absztrakt állapotának a módosítását
  - ez történhet az állapotmódosításra történő közvetlen felkérésre
  - vagy bizonyos jelzett eseményekre való közvetett reagálásra
- az entitásnak jeleznie kell az absztrakt állapotának megváltozását a kimenete felé

Aktív entitás esetén adott *hatás felvétele* a következő elemekből épül fel:

- az entitás lekérdezi a bemenete (absztrakt) állapotát
- a bemenet és a saját absztrakt állapota alapján esetlegesen állapotot vált

- ha történt „lényegi” állapotváltás, akkor ezt közölheti („jelezheti”) a kimenetén lévő entitásokkal

Aktív entitás esetén adott *hatás érvényesítésének* az elemei a következők:

- vagy közvetve:
  - közli az (esetlegesen megváltozott) absztrakt állapotát, mint egy külső eseményt,
  - majd a függő entitás teszteli a környezetét és elvégzi az esetleges módosítást
- vagy közvetlenül:
  - az entitás a saját állapota és a függő entitás saját és környezetének állapota alapján közvetlenül végrehajtja a függő entitás módosítását.

A hatás érvényesítésének kétfajta módjában a közvetettség (azaz a szabadság-fok), illetve a beavatkozás erősségében láthatunk különbséget. Az első módszer nagyobb önállóságot és döntési jogkört hagy a függő entitásnak, míg a második egy kifejezett állapot-mederbe terelést kényszerít ki. Az első így variábilisabb, nagyobb változatosságú rendszert eredményez, míg a második szorosabb korlátozásokat képes fenntartani (!).

Passzív entitás esetén az *entitás hatásának érvényesítési* módja a következő:

- az entitás válaszol az absztrakt állapotára vonatkozó kérdésekre

Passzív entitás esetén az entitást ért *hatás felvételének* a lehetőségei:

- közvetve: az entitás fogadhat (külső) eseményeket, melyre
  - a saját és környezete állapotának függvényében esetlegesen megváltoztatja állapotát,
  - esetleges akcióval a kimenetén további hatásokat válthat ki (közvetve egy eseménnyel vagy közvetlenül egy módosítással).
- közvetlenül:

- az entitás lehetővé teszi, hogy kívülről megváltoztassák az állapotát
- ennek következményeként közvetve vagy közvetlenül további hatásokat válthat ki

Az entitások rendszerét így olyan meghatározottságként tudjuk megadni, amely alapvetően adott feltételek mellett bekövetkező állapot-transzformációk rendszere. A közvetettség, a nagyobb szabadságfok több (esemény-) értesítést alkalmaz, nem pedig a másik entitás állapotváltozásának a kikényszerítését. A nagyobb szabadságfok nagyobb variabilitást tesz lehetővé.

(***Entitások egymásrahatásának megadása.*** Az entitásokra felbontott rendszert, azaz az egymáshoz kapcsolódó entitások rendszerét így egy hármas eszközkészlettel adhatjuk meg. Az entitások rendszerében egy-egy entitás:

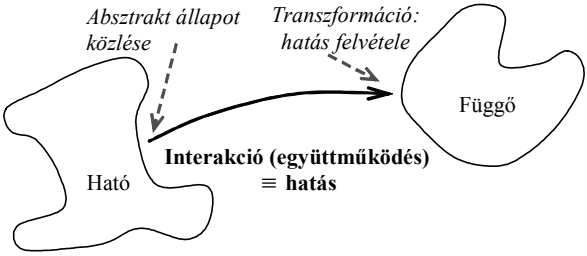
- (esemény) a belső vagy külső eseményének hatására
- (belső-transzformáció) állapotot válthat a saját és környezete állapotának függvényében,
- (akció) melyről értesítheti környezetét vagy kikényszerítheti annak változását

)

Az összekapcsolódó entitások működését, azaz a rendszer entitásai (részei) közötti **interakciót** (más néven **együttműködést**) adott, hatónak nevezett részektől adott, függőnek nevezett részek felé történő információtovábbításként valamint a függő részeken végrehajtott állapot-transzformációként definiáljuk, mely során a ható részek közlik absztrakt kiterjesztett állapotaikat és adott (absztrakt) módon beavatkozhatnak a függő részek állapotaiba, ill. változásaikba. Az interakciók a részek közötti hatásokat reprezentálják, azaz az interakció és a hatás fogalmát ekvivalensnek tekintjük.

(Az interakció a *ható részek oldaláról* tekintve az *absztrakt kiterjesztett állapotuk közlését* jelenti (a rész nem is „ismer” ennél több információt). Adott jellegű rész *absztrakt állapota* ezért értelmezhető *absztrakt hatásként*, amelyben elvonatkoztattunk a függő résztől, illetve annak állapotában okozott változástól.)

(Az interakció a *függő részek oldaláról* tekintve a *hatás felvételét* jelenti, mely esetén megváltozhat a részek állapota. A rész változása legfeljebb önmagára és a kimenetére terjedhet ki. Adott rész esetén *adott jellegű hatás felvétele* egy *absztrakt hatás*, melyben eltekintünk a ható résztől, illetve annak állapotától.)



53. ábra. Interakció (hatás).

Tevékenység

Startpontú kvantumrendszerek esetén a változás megadható a kezdő időpont állapotából a következő időpont állapotába, majd ismételten az ezután következő időpontok állapotába váltó transzformációs lépések sorozataként. Magát a transzformációt időtartam nélküli műveletnek tekintjük, ami az adott (kiterjesztett) állapot alapján meghatározza a vizsgált időpontban érvényes állapotot.

Az átmenet összetett transzformációját (függvényét) *elemi transzformációs* lépések sorozatává (eleminek tekintett függvényekből képzett összetett függvénnyé) bonthatjuk. Egy összetett transzformáció végrehajtása közben (például egy transzformáció-sorozat egy közbülső lépése esetén) az entitás, vagy az entitások együttes rendszere csak részlegesen megváltozott, azaz *inkonzisztens állapotban* lehet. Az entitás állapota a transzformáció végrehajtása közben nem értelmezett, így azt fel sem lehet használni a szükséges transzformációk meghatározásához, ezért az entitás ill. entitások az összetett transzformációk végrehajtása alatt nem fogadhatnak eseményeket/módosítókat.

Egy transzformáció például úgy alakítható időtartammal rendelkező műveletté, ha a transzformáció után a rendszer „megvárja” a kvantum-

rendszer következő vagy egy későbbi időpillanatának a bekövetkeztét. Egy időtartam nélkülinek tekintett, ezért félbeszakíthatatlan transzformáció és az azt követő, következő ütemig tartó várakozás együttesét **tevékenységnek** nevezzük. A transzformáció lehet összetett: ebben az esetben az altranszformációk függő vagy független kompozíciójaként áll elő. A transzformáció lehet üres (identikus): tehát a „legelemibb tevékenység" egyszerűen a következő ütemig történő várakozás.

Elemi tevékenységekből, azok sorozataként **összetett tevékenységet** képezhetünk, amely bármely elemi tevékenység üteme után félbeszakadhat, például egy külső esemény bekövetkezésének hatására.

Az entitásnak valamely szigorúan monoton (halmozódó vagy csökkenő értékének) és kizárólag az időtől függő változását, valamint az arra vonatkozó értékkorlátok együttesét **időhatárnak** nevezzük, amennyiben az adott értékkorlát átlépése hatást eredményezhet az entitásra vagy annak környezetére. **Elemi időhatár** egyetlen értékkorlátot jelent.

(Az összetett (több értékkorlátot tartalmazó) időhatár megadható elemi időhatárok sorozataként. *Az elemi időhatár egy adott időpontig tartó várakozásként jelenik meg*, ami valójában a halmozódó vagy fogyó mennyiség szignifikáns megváltozásának (absztrakt értékváltozásának) a képe.)

(Kvantumrendszer esetén a legkisebb időhatár a következő időpontig történő várakozás, mivel a „korábbi" időpontokban a rendszer nincs értelmezve. *Az időhatár lejárta egy eseménynek felel meg,* ahol a bázisállapothoz képest az értékkorlát átlépése jelenti a tényleges (absztrakt) állapotváltást.)

(A tevékenység elkezdődése egy eseményt határoz meg. A tevékenység befejeződése egy eseményt határoz meg. A tevékenységek időtartamhoz kötődnek, mivel várakozást is tartalmaznak, míg a transzformációk „azonnali" műveletek.)

(*A tevékenység egy állapot-kettőst határoz meg:* a végrehajtó entitás a tevékenység megkezdésével a tevékenység végrehajtásának megfelelő absztrakt állapotba kerül, majd a tevékenységből való kilépéssel elhagyja azt az állapotot, ezért egy másik absztrakt állapotba vált. *A tevékenység és az absztrakt állapot fogalmai* ezért *egymásnak megfeleltethetők.*)

A tevékenység és az absztrakt állapotok kapcsolata azt is jelenti, hogy bizonyos entitások változása függhet attól, hogy adott entitás éppen

végrehajt egy adott tevékenységet, vagy hogy már befejeződött-e (esetleg félbeszakadt) egy-egy tevékenysége. A tevékenységek befejeződésének vagy félbeszakadásának (együttesen: „végződésének") eseményéig történő várakozást **szinkronizációnak** nevezzük.

Absztrakt állapot, esemény és módosító

A rendszer adott részére, pl. egy környezetével együtt vett kiterjesztett entitás állapotára végezzük el a következőt: A ténylegesen felvett állapotokat csoportosítsuk véges számú diszjunkt részhalmazokba, osztályokba. Amennyiben még kettő vagy több osztályt kapunk, a következő lépésben bizonyos osztályok összevonásával határozzunk meg egy újabb osztályozást. Legfeljebb véges számú lépésben végezhetjük el a csoportosításokat, mivel az első lépésben véges számú felosztást engedélyeztünk, a további lépésekben pedig csökkennie kell az osztályok számának. A részrendszer állapotait csoportosító osztályok-alosztályok rendszerét a **részrendszer állapotának lépésenkénti véges felosztásának** nevezzük. Ez a lépésenkénti véges választású közelítés módszerének egy egyszerűbb változata az absztrakt állapotok meghatározására.

(*Az állapotok lépésenkénti véges felosztása egy szempont*, amennyiben az állapotokat egy-egy adott időpontban felvett állapotkombinációinak tekintjük. A szempont esetén az *i0* a konkrét állapotokat határozza meg, utolsó $i\infty$ lépésként pedig a ténylegesen felvett állapotok teljes halmazát kapjuk.)

(*Egyetlen, nem a teljes tényleges állapothalmazt megadó absztrakt állapot egy lépésenkénti véges felosztást határoz meg*, ahol az egyetlen köztes szinten az s absztrakt állapotnak megfelelő csoport, illetve a teljes halmaz és az absztrakt állapotba tartozó állapotok különbségének $\neg_s$ csoportja, mint két osztály szerepel.)

Mivel az állapotok több szempont szerinti felosztását is elkészíthetjük, ezért az absztrakt állapotok között bonyolult viszonyok állhatnak fenn:

(Az entitás bizonyos absztrakt állapotai más (kevésbé absztraktabb, azaz pontosabb) absztrakt állapotok gyűjtőfogalmai („szuper-állapotai") lehetnek. Ezért a rendszer adott része esetén egyszerre több, következmény viszonyban álló absztrakt állapot is teljesülhet.)

(Mivel több szempont is meghatározható, ezért a rendszer adott részére egyszerre több (nem feltétel vagy következmény viszonyban lévő) absztrakt

állapot is teljesülhet, mely absztrakt állapotok különböző felosztások eredményei.)

(Minden absztrakt állapothoz tartozó, a felosztás előző lépésében kapott osztályok kizáróak, így ha a (csoportosító) absztrakt állapot teljesül, akkor a felbontás egy absztrakt alállapotának teljesülése a többi osztály nemteljesülését vonja maga után. Ez a kizárásos elv közvetlenül is igaz a lépésenkénti véges felbontás legelső lépéseként kapott osztályozásra.)

Az absztrakt állapotok szoros kapcsolatban vannak a logikai értékű függvényekkel, mivel azok egy-egy kiválasztást határoznak meg.

(***Minden absztrakt állapot kölcsönösen megfeleltethető a tényleges állapotok halmazán értelmezett adott, logikai értékű függvénynek***, ahol az absztrakt állapothoz rendelt függvény igaz értéket ad minden, az absztrakt állapot csoportjába tartozó állapot esetén, egyébként pedig hamisat.)

(A tényleges állapotokon értelmezett logikai értékű függvény alkalmas egy absztrakt állapot definíciójára. Minden, a tényleges állapotokon értelmezett logikai értékű függvény egy egyszerű szempontot határoz meg.)

(*Absztrakt állapotokból a logikai műveletek segítségével további absztrakt állapotokat képezhetünk.*)

Egy kiterjesztett entitás ténylegesen felvett állapotainak halmaza alapján az absztrakt állapotok több, különböző rendszerét is megadhatjuk, azonban mindössze azok a „lényegiek", amelyek befolyásolhatják az entitás, illetve a kapcsolódó entitások működését. A környezete szempontjából az entitásnak azok az absztrakt állapotai a „lényegiek", amelyek hatással vannak a környezet entitásaira; ezeket **publikus absztrakt állapotoknak** nevezzük.

Az entitás ugyanakkor bizonyos belső műveletsorozatot, esetleg párhuzamosan több műveletsorozatot is végezhet. A belső műveletsorozat megadásához is szükség lehet bizonyos absztrakt állapotok azonosítására, például egy adott állapotváltásnak feltétele lehet bizonyos, pl. két tevékenység befejeződését jelző absztrakt állapotok elérése (szinkronizálás). Ezek a **belső absztrakt állapotok** csak közvetve jelennek meg külső állapotként, de alkalmasak az entitás belső változásának absztrakt leírására.

A környezetéből vizsgálva egy adott entitás megváltozása *külső eseményként* jelenik meg, míg az entitás oldaláról ez a saját változás egy belső esemény. A környezet szempontjából ezért elegendő az entitásnak csak azon

absztrakt állapot-változásait megadni külső eseményként, amelyek „lényegiek", azaz a környezetben változást idézhetnek elő, melyeket **publikus eseményeknek** nevezünk. A **belső események** nem feltétlenül eredményeznek publikus eseményt, azonban alkalmasak az entitás belső folyamatainak leírására, azaz az entitás változásának egy absztrakt, elvi szintű megadására.

Az entitások rendszerében az entitások összekapcsolása történhet közvetlen módon: ekkor az állapotát megváltoztató entitás a tőle függő entitásokon közvetlenül hajtja végre az esetleges módosításokat. A nagyobb variabilitást megengedő közvetett mód alapvetően az „értesítések" rendszerét jelenti: a megváltozott entitás eseményként közli állapotának („lényegi") megváltozását; az értesítéseket „figyelő" entitások ennek hatására elvégezhetik a szükséges állapotmódosításokat. A közvetett módnak lehetővé kell tenni a ható-függő entitások összekapcsolását, „összeszövését".

A tiszta átmeneti függvénnyel megadott rendszer esetén az *entitást* a *környezete szempontjából* a következő módon írhatjuk le:

- publikus absztrakt állapotai, amelyek a környezet entitásaira passzív módon is hathatnak (az absztrakt állapotok lekérdezhetők).
- publikus absztrakt eseményei, pl. publikált (más elemek felé közölt) absztrakt állapotváltásai, amelyek aktív módon hathatnak a környezetére
- a kívülről beállítható absztrakt állapotok; a környezet ható entitása a módosító segítségével közvetlenül kiválthatja az entitás állapotának megváltozását
- „figyelő" absztrakt állapot-transzformációk: az entitás ezeknek a segítségével felveheti a többnyire eseményekként megjelenő hatásokat

Az *entitás belső működése* további elemek megadását igényli:

- belső absztrakt állapotok
- belső absztrakt állapot-transzformációk
- tevékenységek

- belső absztrakt események (pl. belső állapotváltások jelzése, időhatárok túllépése vagy tevékenység befejezése…)

Az entitások közvetett összekapcsolása így történhet az absztrakt esemény hatására történő absztrakt transzformációk meghatározásával.

Vezérlés: működés algoritmikus környezetben

Valós világ objektumai esetén az állapotok változása és az objektumok egymásra hatása *automatikusan*, a világ saját törvényei alapján meghatározott módon zajlanak le. Ha a levegő hőmérséklete fagypont alá csökken, befagy a tó vize, a viharos erejű szél ágakat törhet le, stb.

Ha az objektumokat egy algoritmikus környezetben (egy Turing-jellegű gépen, például számítógépen) reprezentáljuk, akkor a törvények automatizmusát szimulálni kell az algoritmikus környezet által megkövetelt módon. Az algoritmusok jellemzője, hogy valamely cél elérésének módját adják meg, azaz bizonyos irányultságon, *szándékosságon* alapszanak. A törvények automatizmusát ezért algoritmikus környezetben szándékossággal kell szimulálnunk. Példánkban a levegőnek közölni kell hőmérsékletét a tóval, amely adott érték alatt „befagyott” állapotba vált át. Másik megoldásként a levegő az adott érték alá csökkent hőmérséklet esetén utasítja a tavat a befagyásra. Lehetséges azonban az irányultság megfordítása is: a tó is lekérdezheti a levegő hőmérsékletét és „dönthet” a befagyásáról. Az első és harmadik változat a valósághoz közelebbi modellt ad, a második viszont hatékonyabban működik, mivel közelebb áll az algoritmikus környezet sajátosságaihoz.

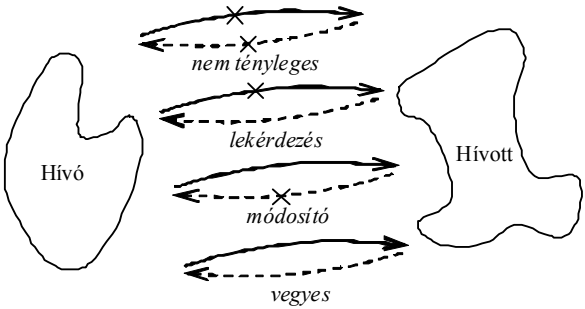
A szándékosság eszköze a kibernetika egyik legalapvetőbb fogalma, a *vezérlés*, amely eredménye a vezérelt dologra kifejtett hatás. A vezérlés egy asszimetrikus kapcsolatot jelent: a vezérlő *irányítását* a vezérelt dolog felett.

Algoritmikus környezetben adott hatást kétfajta módon reprezentálhatunk. Hatást vagy a vezérlő fejt ki a vezérelt objektumra, vagy az objektum lekérdezi a rá ható objektum (absztrakt) állapotát és végrehajtja a szükséges állapotváltást.

A következőkben a hagyományosnak tekinthető függvényhívás/visszatérés vezérlési szerkezettel megvalósítható interakciókat vizsgáljuk. Ebben az esetben a hívó a vezérlő, amely a vezérlést átadja a hívott

programrésznek, amely a vezérelt. A hívó és a hívott részt is működőnek tekintjük, mivel a hívó nem szakította félbe, hanem csak felfüggesztette működését.

A hívás-visszatérés vezérlési szerkezet esetén a hívott adott műveletét (alprogramját: azaz függvényét vagy eljárását) tekintve a vezérlő és a vezérelt objektum között a következő interakciós lehetőségek léteznek:



54. ábra. Hívás-visszatérés interakciók lehetséges típusai.

- A művelet nem ad vissza értéket és mellékhatást sem fejt ki (a várakozást is mellékhatásnak tekintjük). Ez valójában egy üres (identikus) transzformáció, amely nem tekinthető tényleges interakciónak.
- A művelet nem fejt ki mellékhatást, de visszaad valamely értéket. Az objektum csak önmaga és környezete (bemenete) állapotát ismeri, így ez az interakció-típus az absztrakt kiterjesztett állapot *lekérdezését* jelenti. Ez egy egyirányú, a hívottól a hívó felé történő információ-továbbítás.
- A művelet nem ad vissza értéket, de mellékhatást fejt ki. Az ilyen típusú művelet a *transzformáció* vagy *módosító*. Ez egy egyirányú, a hívótól a hívott felé történő információ-továbbítás.
- A művelet mellékhatást fejt ki és értéket is utal vissza. Ez a változat a módosító és a lekérdezés kombinációjaként is tekinthető, amely egy kétirányú információ-továbbítás, egy információcsere. Mivel a visszautalás egyetlen, nem ismétlődő érték meghatározását jelenti, ezért ez a *vegyes* változat megadható egy

kezdő módosító, egy lekérdezés és egy záró módosító elem-hármasával, melyek közül a lekérdezés határozza meg a visszaadott értéket, valamint a kezdő vagy a záró módosító közül az egyik elhagyható.

VI.

Belső rendszer

A belső rendszer, struktúra

Szemléletünkben a rendszer a folyamatot, a változás teljességét jelöli; a rendszeren kívül nincs további külvilág vagy „külső rendszer", amellyel a rendszerünk kapcsolatban lenne. Eddigi vizsgálatainkban a rendszer egyetlen entitását emeltük ki és annak viszonylatában elemeztük a rendszer változását. Ez a módszer kiterjeszthető úgy, hogy a teljes rendszer változását egymásra ható, együttműködő entitások meghatározottságaként adjuk meg. A *belső rendszer* vizsgálatához az entitások egymáshoz való *viszonyát* kell meghatároznunk.

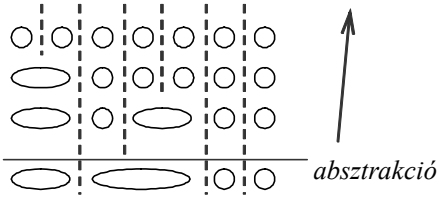
Egy absztrakció eredményeként módosulhatnak a rendszeren belüli hatások. Esetenként látszólagos hatások („babonák") is megjelenhetnek. Ha az absztrakció nem bővíti az elemek változásának tartományait, azaz ha *tartomány-absztrakciót* alkalmazunk, akkor legfeljebb gyengülhetnek, esetenként el is tűnhetnek a hatások, azaz ekkor csökkenhetnek a függések. Egy tartomány-absztrakció arra is alkalmas, hogy „körvonalazzuk" a rendszer egy adott helyéhez tartozó entitását.

A részek közötti kapcsolatok szempontjából tehát különleges szerepűek a tartomány-absztrakciók. Az $i\infty$ kivételével tartományabsztrakciókat tartalmazó szempontot **tartomány szempontnak** vagy **strukturálási szempontnak** nevezzük.

A hatások tranzitívak, így a kapcsolódó részeket diszjunkt „szigetekbe" csoportosíthatjuk, ahol az egymáshoz nem kapcsolódó, független részek különböző „szigeteken" helyezkednek el. Egy tartomány-absztrakció csökkenti a függéseket, így a szigeteken belül újabb zárt csoportokat tudunk azonosítani, mely csoportok elemei (ezen az absztrakciós szinten) függetlenek más csoportok elemeitől. Újabb tartomány-absztrakcióval újabb

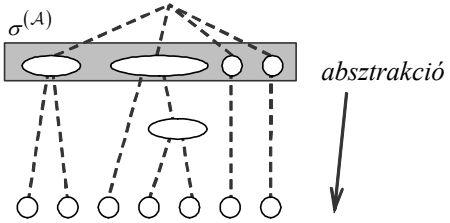
alcsoportokat határozhatunk meg, és így tovább… Mind a „szigetek”, mind azok csoportjai és alcsoportjai a rendszer entitásaiként jelennek meg.

(***Strukturálási szempont a rendszer hierarchikus felbontását állítja elő;*** az egy szinten lévő, egymáshoz nem kapcsolódó részek diszjunktak, a szempont nagyobb értékénél egy rész további független alrészekké bontódhat szét.)



55. ábra: Strukturálási szempont alkalmazásával a rendszer független entitásokra bomlik.

Egy tartományabsztrakció eredményeként kapott részek diszjunkt halmazait mint felbontásokat az eredeti rendszerre visszavetítve egy tartalmazási hierarchiát kapunk, amelyet a rendszer **tartomány-absztrakció szerinti dekompozíciójának** nevezünk. Egy strukturálási szempont szerinti felbontás eredményeként kapott hierarchikus entitás-alentitások rendszerét a rendszer adott **szempont szerinti struktúrájának** nevezzük.



56. ábra: Struktúra, mint hierarchikus dekompozíció.

A **szempont szerint strukturálhatónak** nevezzük a **rendszert**, ha a strukturálási szempont alkalmazása esetén csökkennek a függőségek, azaz a szempont valamely köztes absztrakciós szinten egy korábbi részt kettő vagy több alrészre bont fel.

Tipizálás, fogalmi rendszer

Egy strukturálási szempont absztrakcióit követve a rendszer egymástól független részeire hullik szét, illetve egy rész további független alrészeire bontható. Ugyanakkor, egy (tetszőleges, nem csak strukturális) szempont mind magasabb absztrakciói során az eredeti rendszer hasonló, de kezdetben még különböző részletei közül egyre több sorolható azonos csoportba. A csoportok száma egyre csökken, miközben a rendszer mind nagyobb részleteit foglalják magukba. Az egyre átfogóbb típusok szintén egy hierarchikus rendet alkotnak, mégpedig az absztrakciósorozat végrehajtásának ellenkező sorrendjét követve. A hierarchia legalján a kiindulási rendszer részei helyezkednek el.

A fogalom koncepciója, illetve a pozícionálás technikája lehetővé teszi, hogy egy rendszer különböző részeinek azonos jellegét megfogalmazhassuk: Két (hasonlítható) rendszert **fogalmilag azonosnak** tekintünk, ha ez egyik megfelelő pozícionálásával a másikkal azonosnak tekintett rendszert kapunk (jelölés: $\sigma_A \widetilde{\Leftrightarrow} \sigma_B$). Ha egy σ_A fogalomnak létezik olyan pozícionálása, amely σ_B feltétele, akkor azt mondjuk, hogy a σ_B a σ_A **fogalmi következménye** ($\sigma_A \widetilde{\Rightarrow} \sigma_B$). A σ_B **absztraktabb fogalom** a σ_A fogalomnál, ha annak fogalmi következménye, de fogalmilag nem azonosak.

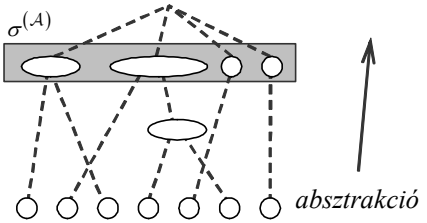
A fogalmi azonosság segítségével az eltérés, illetve a pontosság definíciói a fogalmakra is kiterjeszthetők: Egy rendszer fogalmainak adott szempont szerinti **fogalmi eltérése** a szempont legkisebb olyan értéke, melyhez tartozó absztrakciót a fogalmakra alkalmazva azok fogalmilag azonossá válnak. A fogalmak adott szempont szerinti **fogalmi pontossága**, vagy más néven azok **hasonlósága** az így kapott absztrakt kép.

(Rendszer adott fogalmainak fogalmi pontossága (más néven fogalmi hasonlósága) is a rendszer fogalma. Rendszer adott fogalmainak fogalmi pontossága a vizsgált fogalmak fogalmi következménye.)

Egy rendszer σ_A, σ_B és σ_C fogalmai (például egy kompozit részei) esetén adott szempont szerint a σ_A és σ_B **hasonlóbb fogalmak** a σ_C fogalomnál, ha a σ_A és σ_C, valamint σ_B és σ_C hasonlósága egyaránt absztraktabb fogalom σ_A és σ_B hasonlóságánál.

Amennyiben a rendszer részeit fogalmakként tekintjük, azok egyre absztraktabb képei egymással mind hasonlóbbakká válnak, egyre több lesz közöttük olyan, hogy az egyik absztrakt kép egy pozícionálással valamely másik absztrakt képnek megfeleltethető. A szempont így egy absztrakciós hierarchiát határoz meg, ahol adott absztrakciós szinten azonos csoportba sorolhatjuk a fogalmilag azonossá vált részeket.

Adott szempont esetén egy rendszer diszjunkt részeit tekintve **(fogalmi) tipizálásnak** nevezzük a részek egyes absztrakciós szinthez tartozó képeinek csoportosítását a fogalmi azonosság alapján. (A tipizálás eredménye egy hierarchikus csoportosítás.)



57. ábra: (Fogalmi) tipizálás.

A tipizálás a strukturáláshoz hasonlóan ugyancsak a részek hierarchikus csoportosítását határozza meg. Strukturálás esetén független részeire hullik szét a rendszer, míg tipizálásnál a magasabb absztrakciók egyre bővebb csoportokat, egyre absztraktabb fogalmakat határoznak meg. A szempont egyre erősebb absztrakciói a rendszert egyszerre bontja fel és gyűjti egybe.

Hasonlítható $\sigma_{i\in I}$ rendszereken (azaz $\Sigma^{(\sigma_{i\in I})^*}$) adott szempontok absztrakcióinak alkalmazása esetén vegyük a kapott eredményrendszereket, melyeket a fogalmi azonosság erejéig egy-egy ekvivalencia-osztályba tartózóaknak tekintünk. A kapott fogalmak halmazát a $\Sigma^{(\sigma_{i\in I})^*}$ rendszerek adott szempontok szerinti **fogalmi rendszerének** nevezzük.

(Mivel a rendszer fogalmai hasonlíthatók, így a hasonlítható rendszerek fogalmai is hasonlíthatók, ezért a fogalmi rendszer fogalmai is hasonlíthatók. Egy fogalmi rendszer valamely eleme nem feltétlenül minden rendszer fogalma, mivel fogalmat a rendszer tényleges fogalmából (pozícionálással) képzünk.)

Mivel a fogalmak, illetve azok pozícionálása, azaz alkalmazása is hasonlítható rendszereket eredményez, ezért azokból (például konjunkcióval vagy diszjunkcióval) makrorendszerek képezhetők. Azaz a rendszer bizonyos részeit (amelyek szintén fogalmak) leírhatjuk elemibb fogalmak segítségével.

Hatás ellensúlyozása, kohézió

Egy strukturálási szempont alkalmazása során bizonyos részek hamarabb, mások pedig később válhatnak el egymástól. A szempont így alkalmas a részek összekapcsolódási erejének, azaz a kohéziós erőnek a megadására.

Rendszer adott részei közötti adott strukturálási szempont szerinti **kohéziós erő** a szempontnak az a legkisebb olyan értéke, mely absztrakciójánál a részek már függetlenek. (Független részek közötti kohéziós erő minimális (azaz $i0$). Adott strukturálási szempont esetén a nagyobb kohéziójú részek később válnak szét.)

A definíció alapján így már beszélhetünk egy rendszer adott szempont szerinti *szorosabban vagy gyengébben kapcsolódó részeiről*.

Vegyünk egy entitást, amelyre hat egy másik entitás! A függő entitás eredeti és a hatás eredményeként módosult állapota közötti eltérés egy megfelelő absztrakció végrehajtása során „eltűnhet”, azonossá válhat. Azaz az absztrakció elnyelheti, „ellensúlyozhatja” az entitást ért hatást. Másik hatás esetén esetleg gyengébb vagy erősebb absztrakció szükséges a hatás „ellensúlyozásához”, azaz a végrehajtandó absztrakció erőssége alkalmas az entitást ért hatás mérésére.

Rendszer entitását ért **hatás** adott szempont szerinti **erőssége** a szempont legkisebb olyan értéke, mely absztrakciójánál az entitás és annak a hatás eredményeként módosult változata azonosnak tekinthetővé válik. Ennek megfelelően beszélhetünk adott szempont szerinti erősebb, illetve gyengébb hatásról. (A hatás erőssége az eredeti és a hatás eredményeként megváltozott működések, mint hasonlítható rendszerek eltérése.)

(***Hatások kiterjesztett valós szempont szerinti erőssége távolság*** az absztrakt megadású rendszer egy rögzített entitásának különböző paraméterekhez tartozó működésein.)

A hatás erősségét tehát az eredeti és a módosult változás eltéréseként értelmezzük, azaz a nagyobb eltérés nagyobb hatást jelent.

Rendszer egy entitásának állapotváltozásai, azaz eseményei hatást eredményezhetnek az entitás környezetében. Azaz: az entitás konstans változásához tartozó rendszer-működéshez képest eltérhet a rendszer működése. Az **esemény** adott szempont szerinti **fontossága** vagy **lényegessége**, **szignifikanciája** a szempont legkisebb olyan értéke, mely absztrakciójánál a két működés azonosnak tekinthetővé válik. Ennek megfelelően beszélhetünk **külső esemény** adott szempont szerinti **fontosságáról**, és fontosabb, illetve kevésbé fontos eseményről: Egy entitás szempontjából vett külső esemény adott szempont szerint fontosabb, ha a szempont szerint nagyobb eltérést okoz az entitás változásában.

Az esemény, illetve külső esemény fontossága esetünkben tehát nem annak gyakoriságával vagy ritkaságával van összefüggésben, hanem, hogy az mekkora hatást eredményez.

Ahogy a definíciók is mutatják, a hatás erőssége és a kohézió fogalmai közös alapkoncepcióra épülnek. Ezt úgy is elképzelhetjük, hogy egy entitást ért bizonyos hatások megszüntethetnek az entitás valamely részei közötti kapcsolatot – mégpedig (adott szempont szerint) a szorosabban kapcsolódó részek leválasztásához erősebb hatás szükséges.

Például tekintsünk egy szobát, amelyben két szék közvetlenül egymás mellett, és egymást érintve helyezkedik el. Ha megfelelő irányba megtoljuk az egyik széket (hatás), akkor a két szék együtt mozdulhat el. A szoba többi része független, míg a két szék között van valamekkora kohéziós erő. Ha ellentétes irányba toljuk el a vizsgált széket, akkor az elmozdulása független lesz a másik széktől, a hatás tehát megszüntette ezt a kohéziós erőt. A „szobában van két egymás melletti szék” helyett a „szobában van két szék” absztraktabb kép ellensúlyozni tudja ezt a hatást. Ha egy nagy erőt fejtünk ki a székre, akkor az széttörhet. A törés valószínűleg az illesztések mentén történik, így letörhet a szék lába vagy a karfája. A hatást még nagyobb absztrakcióval kellene ellensúlyoznunk, pl. „a szobában két szék alkatrészei vannak”. A szék lábának egyes részei szorosabban kapcsolódnak egymáshoz, mint a szék többi részéhez, azon belül nagyobb a kohéziós erő, így sokkal erősebb hatás szükséges a részeinek szétválasztásához. Ez a „dekompozíciós” tevékenység-sorozat (azaz a székek szétverése) a rendszer adott szem-

pont szerinti struktúráját adja meg: *székek(szék(karfa, háttámla, ülőke, láb1, láb2, láb3, láb4), ...)*.

Implicit és explicit kapcsolat

Rendszer két entitása közötti kapcsolat az egyik entitás hatását jelenti a másik entitásra és/vagy a másik hatását az elsőre. A hatás a függő entitás működésének (változásának) megváltozásaként jelenik meg, mégpedig feltételes („ha-akkor") módon: azaz *ha* a ható entitás adott módon működik, *akkor* az a függő entitás működését adott mederbe tereli.

A hatás bizonyos együttes változásként jelenik meg a külső szemlélő számára, azaz a ható entitás változására a függő is megváltozik.

Ha a kompozit rendszer entitásával fogalmilag azonos entitás a rendszer valamely más helyén is megjelenik, akkor a közös környezetük adott hatására mindkét entitás azonos módon reagálhat. A külső szemlélő számára mindez úgy jelenik meg, hogy az entitások szinkronban változnak, egyszerre módosulnak. Látszatra az entitások kapcsolatban vannak egymással, holott a valóságban lehetséges, hogy azok között nincs is tényleges kapcsolat. A ható „külső entitás" szempontjából a két entitás között létezik valamiféle kapcsolat, hiszen az általa kifejtett hatásra azonos módon reagálnak. Ugyanez mondható el, ha a két entitás azonos módon hat valamely „külsőként" tekintett entitásra.

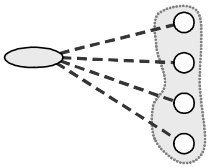
(***Kapcsolat, mint a rendszer következménye:*** Adott entitások közötti adott kapcsolatok a rendszer következményét határozzák meg, amely az entitások állapotait a hatások és azok eredmény-korlátozóinak pontosságáig adja meg, valamint eltekint a rendszernek a kapcsolódó entitásokon kívüli részeitől. Tehát: ***a hatás a rendszerre vonatkozó információ***, mivel hatás esetén a korlátozó nem lehet a tetszőlegesség.)

(***Hasonló entitások azonos módon kapcsolódnak más entitásokhoz.*** A (fogalmi szinten) hasonló entitások azonos jellegű módon kapcsolódnak egymáshoz vagy más „külső" entitásokhoz, amennyiben a fogalmi hasonlóság szempontjának alkalmazása nem módosítja a hatást.)

(Az előző kijelentés megfordítása: ***Egy entitáshoz azonos módon*** (ható és/vagy függő) ***kapcsolódó más entitások fogalmi hasonlóságot határoznak***

meg, ahol a hasonlóság szempontját a kapcsolatok által meghatározott absztrakció adja meg.)

Entitások fogalmi hasonlósága esetén azok azonos módon kapcsolódhatnak más entitásokhoz, azonos jellegű hatást fejthetnek ki vagy azonos módon reagálhatnak azok hatásaira, amelyek így a külső szemlélő számára kapcsolatban lévőknek tűnnek, melyet **implicit kapcsolatnak** nevezünk. Az entitások közötti tényleges kapcsolatot megkülönböztetésül **explicit kapcsolatnak** is nevezzük.



58. ábra: Implicit kapcsolat, mint fogalmi hasonlóság.

Cseréljük ki a rendszer egy entitását valamely másik entitásra! Fogalmilag azonos entitások esetén a különbség észre sem vehető. Fogalmilag különböző entitások esetén a működés megváltozása egy megfelelő absztrakcióval ellensúlyozható, mely absztrakciónak az entitás kapcsolatainak megváltozását kell eltűntetnie, így az entitás más jellegű állapotváltozásait, illetve azok más jellegű hatásait.

*(**Kompozit entitását hasonlóbb entitással helyettesítve a rendszer változása kevésbé fog eltérni az eredeti változástól*** a fogalmi azonosság szempontja szerint. Mindez a ***helyettesíthetőség elvéhez*** vezet el: előre rögzített absztrakció esetén egy entitás másik entitással való helyettesítésekor a működések absztrakt képei azonosnak tekinthetők maradnak, ha a két entitás absztrakciója fogalmilag azonos, azaz a két entitás fogalmi hasonlósága megfelel az absztrakciós szintnek.)

Együttműködés

Egy entitás bizonyos eseményei hatást fejthetnek ki valamely más entitásra, amely hatás közvetve további és további entitásokra terjedhet át. A ható entitás szemszögéből mindez úgy tűnik, hogy az entitások együttesen reagál-

nak az eseményre. A külső entitás szempontjából az összekapcsolódó, összefüggő entitások együttesen, egyetlen egységként, egy *makroentitásként* jelennek meg.

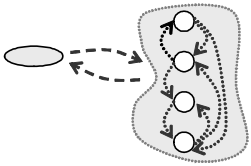
Az együttes reakció összehangolt működést jelent, mely összehangoltság valójában a makroentitáson belüli kapcsolatokat jelenti. Ilyen makroentitás esetén, mivel léteznek belső kapcsolatok, ezért létezik bizonyos belső kohézió. És fordítva: a belső kapcsolatok erőssége határozza meg a makroentitás stabilitását, hogy az mennyire képes ellenállni a külső hatásoknak.

A belső kapcsolatokkal rendelkező makroentitás természetesen több, mint a részek egyszerű összege, és ezt a többletet pontosan a kapcsolódási módok adják, illetve ezek teszik lehetővé a környezet felé történő szervezett akciót és reakciót. Az együttműködő entitások megoszthatnak egymás között nagy külső terheléseket (hatásokat), illetve együttesen nagyobb hatások kifejtésére képesek.

Azon entitásokat, amelyek a belső kapcsolataik révén összehangoltan működnek együttesen **szervezetnek**, összehangolt működésüket pedig **együttműködésnek** nevezzük. Az összehangolt működés azt jelenti, hogy az entitások a belső kapcsolatrendszer alapján közvetlen vagy közvetett módon együttesen fejtenek ki hatást a külvilág felé, illetve együttesen reagálnak bizonyos külső hatásokra.

(A szervezet egy makroentitás, a megfelelő környezettel, illetve külvilággal. Az együttes akciók/reakciók miatt a szervezet olyan egyetlen és egységes entitásként jelenik meg a környezete felé, amely a környezet entitásaira adott módon hat, illetve azok hatásaira adott módon reagál.)

(A szervezetnek nem lehet része olyan entitás, amely minden további entitástól független. Ezért: *A szervezet részei kapcsolódnak egymáshoz.*)



59. ábra: Szervezet és együttműködés.

(Makro)**entitások bezárásának**, illetve **fogalmak bezárásának** nevezzük azok olyan absztrakt képeit, amely egy csoportba sorolja a környezet felé azonos módon kapcsolódó entitásokat, illetve fogalmakat és azokhoz egyetlen absztrakt képet rendel.

(*A bezárással eltekintünk az entitás pontos belső felépítésétől,* azt „fekete doboznak" tekintjük. *A bezárás az entitás absztrakt modellje,* mivel az a megfelelő függvényekkel megadható. *A bezárás az entitás korlátozója, valamint az entitásra vonatkozó információ nem független entitás esetén.*)

(***Helyettesíthetőség és bezárás**:* Ha az entitás lecserélődik, helyettesítődik egy olyan entitással, amely bezárása megegyezik az eredeti entitás bezárásával, akkor a módosított rendszerben az entitáson kívüli részek (azaz a környezet és a külvilág) működése azonos marad.)

A szervezet külső vagy belső események hatására bonyolult belső tevékenységsorozatokat végezhet el, mely tevékenységeket a belső entitások tevékenységei, valamint a közöttük lévő összetett kapcsolatrendszer valósít meg. A kapcsolatok biztosítják az összehangoltságot, azaz a szervezettséget, így az egyes entitások tevékenységei és akciói, valamint a külső eseményekre történő reakciói egységesen jelennek meg. Amennyiben bizonyos résztvevők helyére megfelelő, fogalmilag hasonló entitást helyettesítünk, akkor a változtatás ellenére a szervezet belső működése ugyanaz marad, így a környezete felé is ugyanazt a külső képet mutatja (mindkét rendszer bezárása azonos).

Egy makroentitáson belüli entitás bezárását ***szerepnek*** nevezzük. (A szerep az entitás absztrakt modellje. A szerep az entitás korlátozója, valamint arra vonatkozó információ, ha az entitás nem független.)

Egy makroentitásban több entitás is betöltheti ugyanazt a szerepet.

(***Szerepre vonatkozó helyettesíthetőség:*** Egy makroentitáson belül, ha egy entitást lecserélünk egy ugyanolyan szerepű entitásra, akkor a makroentitás bezárása nem változik.)

(Érdemes egy rövid megjegyzést tenni a szervezettel, illetve a stabilitással és a szabadsági fokkal kapcsolatban. A szervezet egységesen képes fellépni a környezete felé, illetve egységesen reagál annak hatásaira. A szervezet ezért adott struktúra fenntartását igényli, amely a résztvevők oldaláról

adott szerepek felvállalását jelenti. A szerepek ugyanakkor korlátozók, amelyek csökkentik mind a szerepet felvállaló entitás, mind a szervezet variabilitását, szabadsági fokát. A szervezet stabilitása tehát mind a szervezet, mind résztvevőinek a berögződéséhez, kötött struktúrájához vezet*het*, amely eredményeként pl. a szervezet nem lesz képes valamely új hatásra a megfelelő módon reagálni. A kötöttség feloldható, ha a szerepek könnyen átjárhatók, ami belső dinamizmust jelent – így a kötöttség/szabadságfok (de a külső stabilitás is) akár mérhető is! (szervezetmodellezés?!) –, s ez tud egyszerre a külvilág felé stabilitásként és dinamizmusként megjelenni.)

Egy makroentitás azonos szerepű entitásait **kollekciónak** vagy **gyűjteménynek** nevezzük.

(A kollekció egy makroentitás. Kollekció egy eleme felcserélhető annak egy másik elemével, a kollekció szerepének, a makroentitás belső működésének és bezárásának változatlansága mellett.)

Kollekciók esetén tükröződik az explicit és implicit kapcsolatok kettőssége. Mivel a kollekció elemei azonos szerepűek, ezért bizonyos fogalmi hasonlósággal is rendelkeznek. Ezért úgy is tekinthetők, mint amelyek együttesen töltenek be egy szerepet, de úgy is, mintha a szerep többszöröződne meg.

Egy makroentitásnak készítsük el azt az absztrakt képét, ahol az (adott dekompozíció szerinti) entitásokat azok bezárására helyettesítjük. A kapott absztrakt képet a **makroentitás absztrakt struktúrájának** nevezzük

(A makroentitás absztrakt struktúrája a makroentitás absztrakt modellje.)

(***Makroentitás megadható relációként****:* A makroentitás absztrakt struktúrája a szerepek szintjéig meghatározza a makroentitás működését. A makroentitás felírható egy $\Re(\rho_i : E_{i_j})$ elem n-essel, ahol az elem n-es $\Re$ neve a makroentitás bezárása, a ρ_i a makroentitás szerepkörei, az $\rho_i : E_{i_j}$ pedig az adott szerepet betöltő entitást adja meg.)

Környezet

A ható entitás megváltozása a függő entitás módosulását eredményezheti. A hatásnak valamilyen úton el kell jutnia (aktív vagy passzív módon) a függő

entitásig. A hatás terjedése lehet közvetlen, vagy valamely közvetítő elem(ek) tranzitivitásán keresztül történő közvetett hatás.

(A kapcsolódó entitások együttesen egy makroentitást határoznak meg.)

(A kapcsolódó entitások makroentitása esetén az egyes entitásoknak a kapcsolódásra vonatkozó bezárása az adott entitás szerepe.)

(Kapcsolódó entitások esetén a kapcsolat a függő entitásokra korlátozóként, illetve az entitásra vonatkozó információként jelenik meg.)

(*Az entitások közötti adott kapcsolat megadható elem n-essel*, ahol az elem n-es neve a kapcsolatra utal, a szerepek pedig az entitások szerepkörei. *Adott jellegű kapcsolat ábrázolható relációként, ahol eltekintünk az elem n-es adott szerepeit éppen betöltő entitásoktól.*)

Közvetlenül kapcsolódó entitások esetén az entitás eléri a hozzá kapcsolódó entitásokat, így azokon végrehajthat transzformációkat, illetve lekérheti más entitások absztrakt állapotait, így az állapotmódosításait azok hatása alapján hajthatja végre.

Közvetve kapcsolódó entitások esetén a közvetítő kisebb jelentőségű, mint a ható és a fogadó fél. Ha a közvetítő entitások szerepétől teljesen eltekinthetünk, akkor ez a kapcsolat olyan, mintha közvetlen kapcsolódás létezne a lényegi entitások között. Például egy falat megvilágító fényforrás esetén általában lényegtelennek tekinthető minden egyes „foton-entitás”, azoknak elegendő csak az együttes viselkedését számításba vennünk.

Közvetve kapcsolódás esetén azonban gyakori, hogy a közvetítő elemek módosíthatnak a hatáson. Ebben az esetben tehát meg kell határoznunk egy közvetítő *közeget*, amely egyrészt felelős a hatások átviteléért, másrészt be is avatkozhat a hatásmechanizmusokba.

Közvetve vagy akár közvetlenül kapcsolódó entitások hatásainak leírására a következőkben ismertetésre kerül egy egyszerű, mégis szemléletes módszer, amely alapja a „környezet” mint makroentitás modellezése.

A kapcsolódó entitások valójában egymás környezetében helyezkednek el. Minden egyes kapcsolat maga is egy makroentitás, valamint a kapcsolatok egy halmaza is – egy bővebb – makroentitást határoz meg.

Adott makroentitást, amely adott korlátozásoknak megfelelő entitásokat tartalmazhat, **környezetnek** nevezzük.

A környezet nem egyszerűen az entitások együttese, hanem önmagában is entitás jellegű, így adott tulajdonságokkal, illetve állapotokkal rendelkezhet, reagálhat bizonyos eseményekre, lehetnek transzformációi, tevékenységei, generálhat eseményeket…

Mindezek mellett a környezet

- az „adott környezetben" elhelyezkedő entitásokat tartalmazza.
- felelős a nem közvetlen hatások átviteléért, azaz *közvetítő közeg* jellegű.
- módosíthatja az egyes hatásokat, beavatkozhat a hatás átvitelébe.

Képzeljünk el egy játékot, amely egy labirintusban játszódik! A labirintus szobákból áll, amelyek között mozoghatnak a játékosok. A szobákban különböző eszközök is lehetnek. Például, ha egy bomba van a szobában, akkor a játékos megsérülhet. A programozás-technika, pontosabban az objektumorientált technológia klasszikus kérdése, hogy a bomba robbanását hogyan valósítsuk meg. Egyik lehetőségként a bomba robbanásának kell beindítania a játékos „sérülés felvétele" műveletét. Másik lehetőségként a játékos folyamatosan lekérdezi a bomba állapotát, s annak robbanása esetén elvégzi önmagán a sérülés miatt szükséges módosítást. A probléma ezekkel a megoldásokkal az, hogy ilyenkor a bomba és a játékos túlságosan is szorosan van összekapcsolva, amely a játék módosítása vagy bővítése esetén már rendkívüli nehézségeket okozhat.

A flexibilitást egy harmadik megoldás biztosítja, amely részben már kilép az objektumorientáltság szemléletköréből, s a környezetek elvére épít. Eszerint, megadjuk a bomba és a játékos környezetét, amely valójában a szoba. A bomba a szobának fogja „jelenteni" a robbanás eseményét. A környezet, azaz a szoba a felelős a hatások átviteléért, így sorra veszi az elemeit és közli velük „nagy nyomás és hőhatás történt ekkor és itt". A hatás átvitelekor a környezet például számításba veheti a két entitás közötti távolságot, vagy egy „varázsszoba" csökkentheti vagy éppen megfordíthatja a hatást. A környezet a hatás átvitelekor értesíti a függő elemeket a hatásról, amelyek tovább módosíthatják azt, pl. egy páncél csökkentheti a sérülést.

Nézzünk egy másik példát! Vegyünk egy síkbeli „egyenes" objektumot. Azt, hogy egy pont illeszkedik-e az egyenesre, objektumorientált szemlélet esetén általában az egyenesnél adjuk meg. Ugyanitt vehetünk fel egy másik

lekérdezést, hogy egy egyenest metsz-e egy másik egyenes objektum. Hol kell azonban megadnunk mondjuk egy egyenes és egy körvonal metszését (vagy érintését, stb.) tesztelő műveletet? Az egyenesnél vagy a körnél? A válasz itt is a környezet meghatározása, amely esetünkben maga a sík – itt érdemes felvennünk a különböző síkidomok kölcsönös elhelyezkedését meghatározó szabályokat.

A környezet tehát alapvetően más entitásokat fog egybe, amelyek között általában (de nem feltétlenül) bizonyos hatások érvényesek; azaz a környezet többnyire ezen entitások együttes környezetének a része.

(A szervezet egy környezet, azaz a környezet lazább fogalom a szervezetnél. Például, egymással közvetlen kapcsolatban nem álló entitások esetén a környezet egyirányban továbbíthat feléjük egy hatást.)

(Minden makroentitás egy környezet. A teljes rendszer egy környezet. Minden entitás tekinthető környezetként.)

A környezet tehát önmagában csak a makroentitásnak egy másik elnevezése. A lényeges az, hogy egy-egy környezet milyen szabályokat, azaz korlátozásokat határoz meg a benne elhelyezkedő elemeire.

A környezetet tehát úgy is tekinthetjük, mint adott szabályok gyűjteményét, mely szabályok az entitásokra, illetve a hatások átvitelére vonatkoznak.

Egy környezet alkotóeleme is lehet egy környezet. Korábbi példánkban a szobák együttesen a labirintust mint környezetet jelentik. Hierarchikus környezetek valójában egy kompozíciós hierarchiát jelentenek, azaz makroentitáson belüli (makro)entitásokat, így ebben az esetben érvényes néhány tulajdonság:

(***Propagáció elve:*** A külső környezet (makroentitás) tulajdonságai és szabályai alapértelmezésként érvényesek az alkörnyezetben, amely azonban át is definiálhatja azokat. A külső környezet „propagálja", azaz felajánlja a rá vonatkozó entitás-tulajdonságokat és szabályokat. Például egy vállalat (mint környezet) az elnevezését, levelezési-, email- és web-címét „felajánlhatja" alapértelmezésként a telephelyeinek, azonban azok más értékeket is meghatározhatnak.)

(***Delegáció elve***: A környezet a fogadott hatásokat „delegálja", azaz továbbíthatja valamely részei (entitásai, alkörnyezetei) felé, illetve egy összetett tevékenységet a részek tevékenységeivel és a részek egymás közötti hatásaival valósít meg. A vállalat például egy projekt felvállalása és

megvalósítása során különböző részegységeinek és entitásainak továbbítja a feladatokat.)

A környezetekre és alkörnyezetekre való felbontás nem feltétlenül a fizikai elhelyezkedést követi. A környezet a kapcsolódó entitások együttes tere, „környezete". Például egy újság előfizetői nem feltétlenül laknak ugyanabban a városban. A példa alapján ugyancsak megfigyelhetjük, hogy a különböző szempontok szerint meghatározott környezetek különböző hierarchiákkal adhatók meg.

A környezetek a változások, a dinamika *helyszíneit* határozzák meg, így egy környezetbe beléphet egy entitás, majd elhagyhatja azt, ahogy egy játékos belép egy szobába, majd átmegy egy másikba.

Strukturálási szintek

A teljes összetett rendszer leírásakor azt összefüggő részeire bonthatjuk fel, azaz strukturálhatjuk, így külön adhatjuk meg a részeket, illetve az azokra vonatkozó együttes szabályokat. A felbontást tartomány-absztrakciók segítségével végezzük el, így a rendszer dekompozícióit kapjuk.

A strukturálásnak több szintje is meghatározható.

Egyetlen tartomány-absztrakció eredményeként kapott részek diszjunkt halmazait **reguláris dekompozíciónak** vagy **reguláris struktúrának** nevezzük.

A reguláris struktúra a részek egymásmellettiségeként határozza meg a rendszert.

A tartomány-absztrakciókat tartalmazó szempont eredményeként kapott részek-alrészek egymásba ágyazott halmazait **hierarchikus** vagy **környezetfüggetlen dekompozíciónak** vagy **struktúrának** nevezzük.

A környezetfüggetlen struktúra a részek-alrészek hierarchiájaként írja le a rendszert, amely egy faként ábrázolható, azaz olyan egy-kezdőpontú irányított gráfként ábrázolható, amely minden más csúcspontjába a kezdőpontból csak egy úton juthatunk el.

A tartomány-absztrakciókat tartalmazó szempont eredményeként a részek-alrészek egymásba ágyazott halmazait kapjuk. Rendeljünk a szempont minden egyes indexértékéhez valamely *módszert* (minden szinthez

ugyanazt a módszert is rendelhetjük). A szinten lévő elemeket ugyanannak az elemnek tekintjük, ha a módszer az elemekhez fogalmilag azonos rendszereket rendel. Az így kapott struktúrát **hierarchikus tipizált** vagy **környezetfüggő struktúrának** (vagy dekompozíciónak) nevezzük.

Minden szinthez a megfelelő identikus módszert rendelve, egy strukturálási szinten azonos elemnek tekintjük a fogalmilag azonos részeket.

A környezetfüggő struktúra egy olyan irányított gráfként ábrázolható, amely egyes élei azonos elemre, azaz azonos típusú elemekre vonatkozhatnak. A környezetfüggő struktúra így olyan egy-kezdőpontú irányított gráfként ábrázolható, amely minden más csúcspontjába több útvonalon is eljuthatunk.

Környezetfüggő strukturálás esetén, amennyiben a szempont összes absztrakcióit végrehajtjuk, akkor utolsó lépésként az entitásokból tetszőleges absztrakt rendszereket kapunk. Azaz ekkor egy hálót kapunk, olyan irányított körútmentes gráfot, amely egyetlen kezdőponttal és egyetlen végponttal (levélelemmel) rendelkezik.

Vegyünk egy környezetfüggő dekompozíciót! A felbontás egy közbülső szintjén lévő entitás számára „lefelé", a dekompozíció irányába tekintve, egy részhálót látunk. Ha a típusokat nem tudjuk azonosítani, akkor olyan, mintha egy fát látnánk, azaz ez megfelel egy környezetfüggetlen dekompozíciónak. „Felfelé", azaz a makrorendszerek irányába tekintve egy bennfoglalási sorozatot, a makrorendszerek egyre bővebb körét kapjuk.

Több strukturálás eredményeként kapott hierarchikus tipizált struktúrákat együttesen **általános struktúrának** vagy **általános dekompozíciónak** nevezzük.

Általános dekompozíció során tehát több (tartomány-absztrakciókat tartalmazó) szempont szerinti felbontást is engedélyezünk. Amennyiben egy dekompozíciós szint adott entitását/entitástípusát vizsgáljuk, akkor úgy tűnik, mintha „lefelé", azaz a dekompozíció irányába egy fát látnánk; ha az azonosságok is megfigyelhetők, akkor ez egy háló, vagy részháló. Mivel az entitás képe több szempont szerinti felbontásban is szerepelhet, ezért „felfelé", a makrorendszerek irányába is egy hálót, vagy az azonosságok figyelmen kívül hagyásával egy fát „látunk".

(***Általános dekompozíciót egy irányított körútmentes gráffal (DAG: Directed Acyclic Graph) adhatunk meg.***)

Az általános struktúrához tartozó DAG, azaz irányított körútmentes gráf általános jellege azt jelenti, hogy a rendszerek vázának ábrázolásához nem szükséges általános gráfot alkalmaznunk, hanem kihasználhatjuk az irányítottság, illetve a körútmentesség előnyeit.

Egy DAG adott szintjén elhelyezkedő entitás esetén az entitás számára „lefelé", a komponensei felé, valamint „felfelé" a makroentitások felé is egy hierarchiát lát, környezetfüggő, esetleg környezetfüggetlen módon (ha nem képes a hasonlóságok azonosítására). „Felfelé" ez az egyre bővülő, különböző szempontok szerinti makroentitásokat jelenti. Gyakran csak az a kérdés, hogy egy entitás egy adott makroentitás része-e, ill. egy entitás beletartozik-e egy makroentitás típusába. Az ilyen kérdésekhez már a teljes hierarchia ismerete sem szükséges; a „felfelé" irányban elhelyezkedő (sorozatként vagy faként megjelenő) entitásoknak elegendő mindössze a halmazát tekinteni.

(Az egyes strukturálási-szintek egyre bővülő fogalmakat jelentenek, tehát egy reguláris dekompozíció egyben környezetfüggetlen, környezetfüggő és általános dekompozíció is, és így tovább. Bizonyítás nélkül említsük meg, hogy ezek *a szintek közvetlen kapcsolatban vannak a generatív nyelvtanok szintjeivel*.)

Szabályrendszer

Rendszerek nagy csoportját írhatjuk le úgy, hogy egy-egy megadás viszonylag kevés „helyet" (pl. rövidebb szimbólum-sorozatot) igényeljen, ha azt „meghatározottságokként" tudjuk megfogalmazni.

Tekintsük a következő módszert: vegyünk adott hasonlítható rendszereket és adjunk meg olyan $c_i \mapsto \sigma_j$ leképezéseket, ahol c_i (a **szabály feltétele**) és σ_j (a **szabály következménye**) egyaránt valamely hasonlítható rendszer egy fogalmának a pozícionálása (amelyek így ugyancsak a rendszerekkel hasonlíthatók). A leképezéseket úgy értelmezzük, hogy ha a rendszerre teljesül a c_i feltétel, azaz a c_i a rendszer absztrakciója, akkor a rendszerre teljesül a σ_j következmény, tehát az is a rendszer absztrakciója lesz (azaz a rendszerre vonatkozó információ, ha a σ_j nem a tetszőleges

absztrakt rendszer). A c_i és a σ_j rendszerek esetén megengedjük, hogy azokat bizonyos argumentumok alapján állítsuk elő, így az $r(a_{p\in P})$ **szabály** a p argumentumok megadásával egy $c_i \mapsto \sigma_j$ leképezést állít elő.

A szabályok megadása esetén jelöljük meg a **tényleges szabályokat**, melyeknél, ha több tényleges szabályhoz tartozó leképezés baloldala is teljesül, akkor a jobboldalaknak együttesen kell teljesülni, így azok nem lehetnek ellentmondók. Valamint, tényleges szabályok esetén megköveteljük, hogy azok következménye nem lehet tetszőlegesség. A nem feltétlenül tényleges szabályt **absztrakt szabálynak** nevezzük. A (paraméterezhető) szabályok együttesét, a tényleges szabályok megjelölésével **szabályrendszernek** nevezzük.

Amennyiben a szabály jobboldala egy tetszőleges absztrakt rendszer, akkor a szabály nem korlátozza a működést (így az nem tényleges szabály), tehát ilyen szabályokkal a szabályrendszer tetszőlegesen kiegészíthető, illetve azok elhagyhatók. Az ilyen szabályokat **tetszőleges absztrakt szabálynak** nevezzük.

A szabályok egy egyszerű következtetést határoznak meg: *„ha”* a baloldal teljesül, *„akkor”* a jobboldalnak is teljesülnie kell. Egy teljesülő szabály esetén azonban mind a baloldal, mind a jobboldal a rendszer egy-egy következményét határozza meg, amelyek ekkor együttesen teljesülnek, azaz a kettő konjunkciója a rendszer következménye. A teljesülő (nem tetszőleges absztrakt) szabály tehát valójában a rendszerre vonatkozó tényleges információ. Így a szabály alkalmazása valójában egy helyzet, azaz egy korlátozó megjelenésének a felismerése, amely „követelmény-”, vagy más néven feltétel-részének megjelenése már utal a következmény, illetve a teljes szabály, mint korlátozó megjelenésére.

(*Tényleges szabály a rendszerre vonatkozó információ,* amennyiben a bal- és/vagy jobboldala nem tetszőleges absztrakt rendszer.)

(***A szabály a rendszer fogalma, a tényleges szabály a rendszer tényleges fogalma.***)

A szabály alkalmazása tehát nem a hagyományos értelemben vett, logikai értékű implikáció, hanem következtetés, azaz a szabályok egymással

való kombinálásának eredményeként kapott újabb állítás. A teljesülő szabály alkalmazása valójában az adott helyen és időben „megjelenő” fogalom felismerése.

A tényleges szabályok a rendszer bizonyos jellegű *szabályosságát* jelentik.

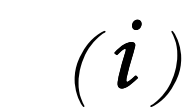
A $\mapsto \sigma_j$ alakú (azaz feltétel nélküli) korlátozó szabályok esetén ez a szabályosság azt jelenti, hogy a rendszer a σ_j korlátozóinak értelmezett időpontjaiban a rendszer nem veszi fel az összes lehetséges állapotot, hanem csak azok bizonyos valódi részhalmazát.

A $c_i \mapsto \sigma_j$ hozzárendelést meghatározó szabályok esetén a „szabályosság” a c_i és σ_j korlátozók között jelenik meg, azaz a c_i teljesülése, más néven: a c_i fogalom megjelenése a σ_j, pontosabban a c_i és σ_j együttes megjelenését is jelenti.

A szabályosságok kiemelt jelentőséget kapnak, ha azok többször is megjelennek a rendszerben, pl. más időben vagy más helyen. Többszörös megjelenés esetén az ismétlődések kiemelhetők és a tényleges megjelenési formák a szabályok alkalmazásaival helyettesíthetők.

A kiemelés csökkentheti az ábrázolás méretét, amennyiben a kiemelt szabály és annak alkalmazásai rövidebben (pl. kevesebb szimbólummal) ábrázolhatók, mint a tényleges működések; pontosan ekkor célszerű végrehajtani a kiemeléseket.

(A szabályok egy *„ha-akkor”* jellegű kapcsolatot írnak le, hasonlóan az entitások közötti hatásokhoz. A szabály a hatás fogalmának általánosítása; ebben az esetben a feltétel és a következmény különböző absztrakciós szinteken is elhelyezkedhet. Ezért ***a rendszeren belüli hatás a rendszer tényleges szabálya***. Szabályok esetén továbbá megengedjük a parametrizálást, valamint a két oldalt bizonyos fogalmak pozícionálásaként határozzuk meg.)



Intelligencia, mint a megértés és az absztrakció képessége

Szemléletünk alapvetően a rendszerek ábrázolásának, illetve a rendszerekre vonatkozó ismeretek finomításának eszközeit tárgyalja. Amennyiben hipotézisként, vagy inkább csak egy lehetséges megközelítési módként elfogadjuk, hogy az „intelligens viselkedés" alapja a helyzetfelismerés, azaz a helyzetre vonatkozó megfelelő ismeret megszerzése, akkor a szemlélet eszközeit felhasználva elgondolkodhatunk az intelligencia természetén. Még egyszer hangsúlyozzuk, hogy nem az intelligencia teljes körű definíciójára törekszünk, hanem mindössze egy *vélemény* továbbgondolására, amely talán majd egyszer segítheti az intelligencia természetének részletes elemzését.

Fogadjuk el hipotézisként a következőket:

*Az **intelligencia** a külsőleg megadott szabályok, vagy a változáson belüli ismétlődések, azaz szabályszerűségek azonosításának a képessége.*

A fentiek alapján nagyobb, vagy ***magasabb intelligenciájúnak** nevezhetjük a megfigyelők közül azt a megfigyelőt, amely a rendszerek adott körére (azaz adott fogalomkörre) vonatkozóan súlyozottan több tényleges szabályszerűséget állapít meg.*

Elemezzük az előző megfogalmazás egyes részeit:

- Az intelligenciát viszonylagosan, azaz más megfigyelőkhöz viszonyítva is értelmezhetjük.
- A *magasabb* intelligenciát csak adott rendszerekre, azaz fogalomkörre vonatkoztatjuk, így más fogalomkör esetén más lehet bizonyos intelligenciák viszonya.
- Több szabályszerűséget kell azonosítani. Erre az egyik lehetőség, ha a magasabb intelligenciájú megfigyelő minden olyan szabály-

szerűséget azonosít, mint bármely más megfigyelő, valamint további szabályszerűségeket is. Az azonosított szabályoknak megadhatjuk egy súlyozott átlagát, mely mérőszám vegyes esetekben is alkalmas az összehasonlításra.

- Fontos, hogy tényleges szabályszerűségek megállapításáról van szó, azaz például a nem-tartományabsztrakció esetén megjelenő hamis hatás azonosítása, a „babona" nem jelent többletet. A tényleges szabályok, illetve általában a szabályok fontosságának meghatározására ugyancsak alkalmazható a súlyozás.

A fenti megfogalmazások valójában definíciók. Azért használtuk a „hipotézis" elnevezést, mivel az intelligencia mind köznapi értelemben, mind a pszichológia szakmai értelmében széleskörűen használt fogalom. „Hipotézisen" munkahipotézist értünk; célunk, hogy segítségével felderíthessük a „valódi" intelligencia bizonyos jellemzőit.

Nézzük tehát, hogy a megfogalmazásból milyen jellemzőket állapíthatunk meg. Fontos, hogy a „magasabb intelligenciát" mindig csak adott területre, adott fogalomkörre vonatkoztatjuk!

Problémamegoldás. Az intelligenciát alapvetően csak a helyzet azonosításának a képességével definiáltuk. Ennek azonban közvetlen kapcsolata van a problémamegoldással. A problémamegoldás egyrészt a helyzetre vonatkozó szabályok, különösen azok feltételrészének az azonosítását jelenti. Jelentheti még a szabályok egymással való kombinálást, azaz a következtetést is. A szabályok együttes értelmezése, alkalmazása azonban még nem feltétlenül nevezhető következtetésnek. A probléma megoldása jelenti továbbá a megtalált megoldási mód alkalmazását is, amelyet azonban már nem sorolunk az intelligenciához; egy megtalált megoldási módot más is megvalósíthat.

Önmagában az intelligencia alapvetően passzív, illetve „aktivitása" mindössze a helyzet felismerésére vonatkozik. A passzivitást abban az értelemben értjük, hogy az intelligencia fogalmát csak a helyzet azonosítására alkalmazzuk, amely még nem jelent a helyzetbe való tényleges beavatkozást, akciót.

Absztrakció, fogalomalkotás. A magasabb intelligencia jobban azonosítja a rendszerben megjelenő ismétlődéseket, szabályszerűségeket, amelyeket kiemelve önálló fogalmakként kezel. A rendszerben való ismétlődések, szabályszerűségek meghatározása és kiemelése valójában az absztrakció művelete.

A magasabb intelligencia jobban azonosít váratlan helyzeteket, mivel arra vagy a korábbi fogalmait tudja alkalmazni, vagy az új helyzet szabályszerűségeiből alkot újabb fogalmakat, s így meg tudja határozni az új helyzet belső összefüggéseit.

A magasabb intelligencia mélyebben hatol be kiterjesztett környezete szemantikájába. A megfigyelőt, mint a rendszer egy entitását tekintve az legfeljebb önmaga és környezete, azaz együttesen: a kiterjesztett környezete változását észlelheti. A változás észlelése általában valamely interfészen („illesztésen", jelcsatornán) keresztül, jelekként történik. A jeleket, mint a kiterjesztett környezet változásának leképezéseit tekintve a megfigyelő azonosíthatja a leképezés módját (azaz szabályait), valamint az annak forrásául szolgáló változás szabályszerűségeit. Mindez azt jelenti, hogy a szabályszerűségek meghatározásával az adott formában, adott „szintaktikával" megfogalmazott jelek értelmét, valamint annak szabályszerűségeit, azaz a jelek szemantikáját fel tudja építeni, mégpedig a szabályszerűségek (hasonlóság, kohézió, struktúra, stb.), és ismétlődések azonosítási képességének a mértékében.

Megismerés. Az előzőek alapján a magasabb intelligenciának több ismerete van környezete változásának szabályszerűségeiről, illetve jobban megismerheti azt.

A magasabb intelligencia azonos változást kisebb méretben (például kevesebb szimbólummal) *fogalmazhat meg*, mivel képes az abban megjelenő ismétlődések és szabályosságok kiemelésére.

A magasabb intelligencia azonos helyen több vagy bonyolultabb változást képes ábrázolni, az előző szabály megfordítása alapján.

Intelligensebb ábrázolásmód. Az előzőek alapján meghatározhatjuk az intelligensebb ábrázolásmód fogalmát, mégpedig olyan megadási formaként, amely a hasonlóságok kiemelésével kevésbé redundáns, így alapvetően kisebb méretű formát állít elő.

Intelligens viselkedés. Ha az intelligencia a következtetésen alapulna, nevezhetnénk-e akkor egy olyan programot intelligensnek, amely nem kombinálja a szabályokat, hanem kizárólag alkalmazza azokat? Mégpedig intelligensebbnek tekintünk egy programot, amely több (lényegi) helyzet esetén a megfelelő módon viselkedik. Ugyanígy egy emberi lény is, kizárólag a neveltetése vagy jelleme alapján „megfelelően" viselkedhet egy adott környezetben, anélkül, hogy közben következtetéseket levonására, vagy a szabályok kombinálására kényszerülne.

Az intelligenciára alkotott megközelítésünk szerint intelligensebbnek nevezhetünk egy programot, ha az a helyzetnek megfelelőbben viselkedik, azaz pontosabban meg tudja állapítani bizonyos helyzetek jellegét és arra alkalmazni tudja a szükséges „megfelelő" viselkedés szabályát.

Túlélés és intelligencia. Az intelligenciára alkalmazott meghatározásunk a túléléssel, illetve az alkalmazkodóképességgel is kapcsolatba hozható. A *túlságosan alacsony* intelligencia nem képes azonosítani a túlélést veszélyeztető helyzeteket, vagy azok megoldási módjait, így biztosan állítható, hogy az az evolúció szempontjából káros. A környezethez való alkalmazkodáshoz, azaz a megfelelő szabályok kialakításához szükséges a környezet szabályainak azonosítása.

A túléléshez kell bizonyos fajta aktivitás is. Korábbi megállapításunk szerint azonban az intelligencia alapvetően passzív, azaz csak a helyzet és esetleg az arra alkalmazandó szabály megállapítását jelenti, de nem feltétlenül annak végrehajtását is.

Próbáljunk meg elképzelni egy „túlságosan magas" intelligenciát! Mivel „az" jobban képes azonosítani az ismétlődő fogalmakat, szabályokat, ezért ez a folyamat szinte a teljes működését igénybe veheti, úgy, hogy akcióra gyakorlatilag nem is marad elég erőforrása. A szabályok és a lehetőségek, azaz a lehetséges javítási módok azonosítása miatt folyamatosan környezete tökéletlenségével és tehetetlenségével ütközik, valamint saját korlátaival, amivel még saját burjánzó ötleteit sem tudja a valóságba átültetni. (Erre láthatunk példát a Csodabogár című filmben – eredeti címén: Phenomenon, 1977, rendezte: Jon Turteltaub, író: Gerald Di Pego, főszereplők: John Travolta, Kyra Sedgwick, Forest Whitaker, Robert Duvall).

Egy kis tautológiával azt mondhatjuk, hogy minden felismerés csak annyiban hasznos, amennyiben azt használhatóvá lehet tenni. Az absztrakt és realizálatlan felismerés így sajnos haszontalan és ebben az értelemben feles-

leges is, mindaddig, amíg valamely véletlen helyzet meg nem teremti a szükségességét.

A túlságosan magas intelligencia éppen ezért evolúciósan káros, és talán pontosan ezért állította be az evolúció a jelenlegi átlagra, a 100-asnak nevezett IQ-ra, a homály jótékony fátylát borítva a világ tökéletlenségére és elfedve a javíthatóság, a világ jobbá tételének sok-sok lehetőségét, hogy figyelmünk elegendő legyen egy-egy apró részlettel foglalkozni.

Jegyezzük meg, hogy a közvetlen evolúciós haszon nem az egyetlen, s nem is a legmegfelelőbb mérőeszköz a hasznosság és éppígy az intelligencia működésének az elbírálására. Mind az egyedek, mind azok szerveződései egy összetett és többszintű „célfüggvénnyel" határozzák meg a hasznosságot, ahol a közvetlen túlélés elsődleges szempontjai mellett a belső életminőség, a felfedezések és az alkotás izgalma, a szépség és harmónia átélése kiemelt szerepet kapnak.

(MoNo)

Modeling Notation – Modellező Jelölés

Az elméleti megközelítés segítségével pontosíthatók a modellező nyelvek szükséges jelölései. Mindez elvezethet – egy paradoxonnal élve – egy olyan Általános DSL *(általános „domain specifikus nyelv")* eszközkészletéhez, amellyel bármely rendszer struktúrájával és működésével kapcsolatos alapinformációk megadhatók.

A következőkben bemutatjuk egy kísérleti modellező nyelv (*MoNo* – Modeling Notation – „Modellező jelölés") alapelemeit és alapkoncepciót. A jelölések többnyire azonosak a jólismert UML-lel (illetve OMT-vel), azonban a MoNo:

- rendelkezik vizuális és szöveges megadási lehetőséggel (a vizuális lehetőség diagramokat jelent, a szöveges megadás egy strukturált formában, pl. XML-ben lehetséges; de van egy angol-alapú (angolhoz hasonlító) szöveges megadási lehetőség is)
- egyetlen „diagram-típusa" van, amelyen az UML összes diagramja megadható; egy rendszerről természetesen több diagram is előállítható, amelyek különböző részeket, ill. azokat részletesebben/vázlatosabban ábrázolják
- egy modellnek automatikusan előállítható a kisebb vagy nagyobb részletezettségi fokú ábrázolása
- felülről kompatibilis az UML eszközkészletével
- egzakt „absztrakt szemantikával" rendelkezik, azaz adott platform kiválasztása és tervezési döntések meghozása után a megadott szoftveralkalmazás létrehozható (generálható), tehát nem csak a rendszer vázlatainak ábrázolására ad lehetőséget, hanem programozási nyelvként is használható.

A következőkben a MoNo legalapvetőbb vizuális jelöléseit mutatjuk be.

A MoNo négy alapelemmel rendelkezik:

- strukturális elemeket – mint a rendszer építőkockáit – alapvetően téglalap jelöli
- az időbeliséget alapvetően ellipszis oldalú téglalap jelöli; ez például az állapot és az aktivitás (más néven: tevékenység) jelölése (tevékenyég és állapot hasonló fogalmak: egy időben elnyúló tevékenység végrehajtása alatt egy adott állapotban vagyunk – amíg „futunk”, addig „futó” állapotban vagyunk)
- az időpontokhoz tartozó elemeket, illetve azokat, amelyeknél eltekintünk azok időtartamától, ellipszis jelöli, így ez az akció (félbeszakíthatatlan művelet) jele, de műveletek csoportja (interfész) is jelölhető így.
- a „leíró” elemekkel részletek vagy megjegyzés adható meg.

60. ábra:MoNo alapelemek.

Általában véve: az elemek megjelenését kis képpekkel (ikonokkal) helyettesíthetjük.

Az elemek felső része a *név-rész* (name-comparment); az alatta következő definíciós rész tartalmazhat szöveges és grafikus részeket, akár váltakozva is. Nagyméretű grafikus rész esetén az elem adatai a balfelső sarokra helyezett külön alakzattal adhatók meg (pl. a következő ábrán a környezet esetén).

A típus lehet *absztrakt*, amit dőltbetű jelez. Az elem lehet az elemen belül *konkrét* (pl. tulajdonság vagy rész), amit aláhúzás jelez. Az elem lehet „*többszörös*”, amelyet jelezhet két, kis eltolással egymásra helyezett alakzat.

Az elem lehet „*hivatkozott*”, amit szaggatott oldal jelez (pl. egy külvilágban történő eseményre vagy történésre is így lehet hivatkozni).

Az elem lehet *globális* (UML „osztály-attribútum” és „osztályművelet”, amit két aláhúzás, vagy a kezdő $-jel jelöl).

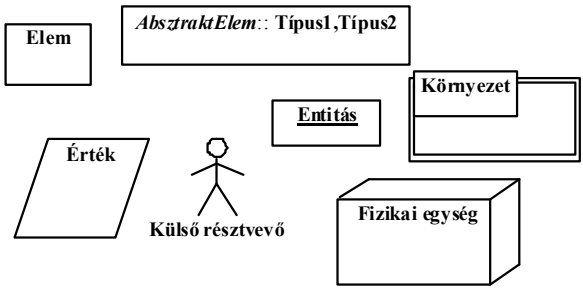
Strukturális elemeket alapvetően téglalapban adjuk meg; ez az entitástípus („*osztály*”) és az entitás („*objektum*”) jelölése (egyetlen entitás („objektum”) önállóan is megadható a típusának („osztályának”) a definiálása nélkül).

Az érték-elem (például egy összetett érték) hangsúlyozható a parallelogramma jelöléssel.

A rendszerhez kapcsolódó külső résztvevő (például egy felhasználó) hangsúlyozható a pálcika jelöléssel.

A „környezet” egy kettősoldalú téglalappal adható meg. A környezet általában egy önálló tulajdonságokkal és műveletekkel rendelkező elem, amely eléri a benne lévő elemeket. (Környezetként adható meg az UML csomag és komponens is).

„Fizikai eszközök” (pl. számítógépek) hangsúlyozása esetén használható a „kocka” jelölés.

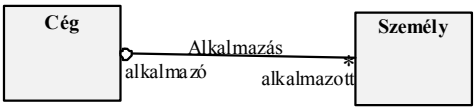


61. ábra:Strukturális alapelemek.

A strukturális elemek különböző módon összekapcsolhatók.

A folytonos (nem szaggatott) vonal a reláció vagy kapcsolat jelzésére szolgál. A kapcsolat végei a szerepkörök, ahol megadható számosság.

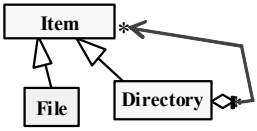
- A számosság *szám* vagy *szám1..szám2* formában adható meg.
- A természetes számokba beleértjük a *„tetszőlegesen sok”* * vagy ● jelét; ez egyben önállóan a 0..* jelölése.
- A többszörösség a „többszörös elem” alakzattal is jelölhető (eltolással egymásra helyezett alakzatok).
- A 0 vagy ○ vagy ? jel az elhagyhatóságot jelzi.
- Az 1 vagy ▌ a „kötelező” jelzése. (A vonal végére helyezett ●▌ így a *„legalább egy, de tetszőlegesen sok”* jele).



62. ábra: Reláció, szerepkörök, számosság

A háromszögben végződő vonal az általánosítás-pontosítás jelzése (a logikai ⇒ implikáció, azaz a *„ha-akkor”* jel változata). Az általánosítás és pontosítás oldalán is megadható számosság – ebben az esetben az általános elem és a változat külön elemként is elérhető (külön vannak tárolva, pl. *Termékleírás* és *Termék* viszonya esetén).

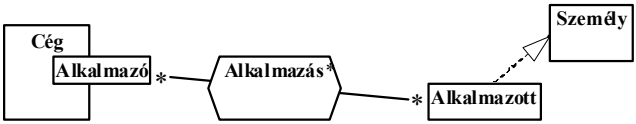
A tartalmazási „rész-egész” viszony („aggregáció” és „kompozíció”) megadható a reláció „egész” szerepkörénél felvett rombusszal, illetve kitöltött rombusszal.



63. ábra: Pontosítás és aggregáció

A reláció önmagában is rendelkezhet mezőkkel és műveletekkel (illetve további relációkkal, stb. – UML *„asszociációs osztálya”*); ebben az esetben a relációt egy (a két odalán csúcsos) hatszöggel jelöljük. A reláció lehet többszörös is (pl. egy Személy ugyazzal a Céggel többféle módon is lehet Alkalmazás kapcsolatban).

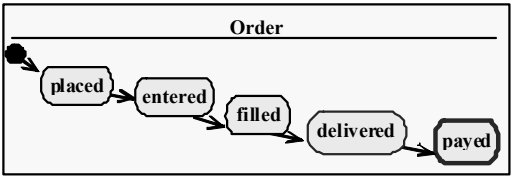
Szerepkörhöz is kapcsolódhatnak adatok, így az is megjelenhet egy speciális (az alapentitáshoz kapcsolódó) entitásként. A szerepkört vagy az alapentitás oldalánál jelöljük (így ez nem az UML „minősítő” jelzése), vagy ahhoz egy háromszögben végződő szaggatott vonallal kapcsoljuk (így ez nem az UML *„realizáció”* jelzése).



64. ábra: (Többszörös) reláció és szerepek önálló jelölései.

A változással kapcsolatos két alapelem a *folyamat* és az *átmenet* jelzése.

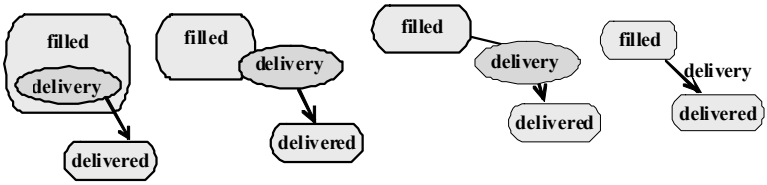
- A *folyamat* egy *időtartamhoz* kapcsolható, így ez az *állapot*, illetve a több lépéses, félbeszakítható művelet, a *tevékenység* jelzése. Grafikus jele az ellipszis oldalú téglalap. (Egy *hosszúidejű folyamat* – pl. egy hosszúidejű együttműködés – entitás-jellegűen jelenik meg, pl. tárolható.)
- Az *átmenet* időponthoz kapcsolható, nem félbeszakítható, vagy olyan művelet(ek) jelölése, amely(ek) esetén eltekintünk az időtől (pl. (tranz)akció). Grafikus jele az ellipszis.



65. ábra: Elem állapotai/állapotváltásai

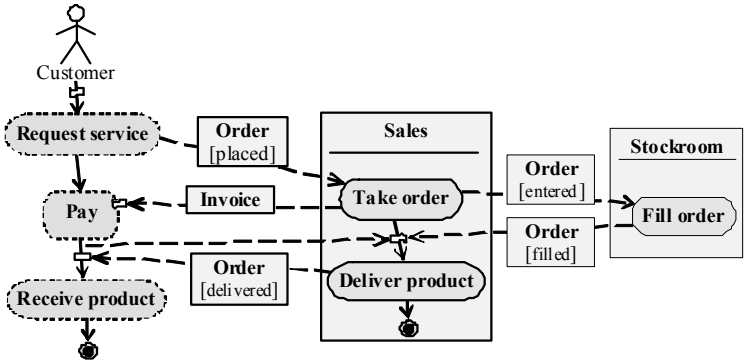
A változás megadható strukturális elemen belül, vagy annak az oldalán, vagy az elemhez kapcsolt vonallal.

- *Publikus* változáselem vastag betűvel hangsúlyozott (állapot esetén: kívülről beállítható, művelet esetén: kívülről meghívható).
- Azonos változáselemek között értelmezett az *általánosítás-pontosítás* viszony (háromszögben végződő vonal), illetve az lehet absztrakt (dőlt betű).
- Állapoton belül jelölt állapot (pontosított) alállapotot jelöl. (Egy strukturális elem egyben önmaga legáltalánosabb, „létező" állapotát is jelöli).
- Műveleten belül további műveletek is megadhatók: ekkor a művelet az állapoton belül értelmezett (vagy azon belül adott módon értelmezett).
    - Belső műveletek a külső oldalán vagy a külsőhöz kapcsolt vonallal is megadhatók.
    - A strukturális vagy műveletelemből kivezető elnevezett nyíl az (átmenetet) indító (ún. „trigger-") művelet rövidített elnevezése.



66. ábra:Állapotból adott eseményre történő átmenet alternatív jelölései

A jelölések felülről kompatibilisek mind az UML életvonalak (lifelines), mind a sávok (swimlanes) jelöléseivel. Az UML életvonal valójában a strukturális elemhez kapcsolódó műveletek sora. Sávok esetén ezek a műveletek az elemen belül vannak megadva (a sáv valójában a tartalmazó elemet jelöli).

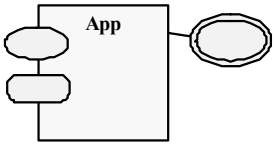


67. ábra: Műveletek

A szaggatott vonalú alakzat a külvilágban történő tevékenységet/eseményt jelzi, ill. olyan feladatot, amelyről a rendszer közvetve (pl. feladatkezelő révén) értesül.

A hullám-alakzat (esetleg csak a ~ jel) a késleltetést (opcionalitást/szinkronizációt) jelöli, azt, hogy a lépés vagy üzenetküldés nem azonnal, illetve nem feltétlenül következik be. Ha a késleltetés alakzatba üzenet érkezik, akkor az üzenet az átmenet vagy művelet feltétele (szinkronizáció).

Műveletek együttese duplavonalú alakzattal hangsúlyozható, így ez alkalmas az interfész (UML „*lollipop*” jelölés), illetve az együttműködés jelölésére is. Az egyszerűbb rajzolhatóság érdekében ezek az alapalakzattal is rajzolhatók. Részletesebb ábrázolás esetén ezek (grafikusan vagy szövegesen) megadhatják a definiált belső műveleteket.



68. ábra: Elem kívülről elérhető műveletei, ill. interfésze

A műveleteket jelölő elemek összekapcsolhatók:

- Névtelen (normál) nyíl: a végrehajtás következő műveletét adja meg. A vonalon a késleltetés (opcionalitás/szinkron) hullám-alakzata jelzi, ha a lépés nem közvetlenül, hanem egy idő után (adott külső körülményeket követően), esetleg csak opcionálisan történik meg.
- Műveletből vagy átmenetből kiinduló, műveletbe vagy átmenetbe érkező *szaggatott* vonalú névtelen nyilak:
    - „Normál” nyílhegy: aszinkron üzenetküldést jelöl, vagy azt, hogy a hívás befejeződésétől eltekintünk („indító”, „trigger” hívás). Ha ez a nyíl egy átmenetre mutat, az azt jelenti, hogy az üzenetküldés kiváltja az átmenetet (ha a fogadó nem a megfelelő állapotban van, az hibát eredményez). Ha a nyíl egy késleltetés hullámalakzatára mutat, akkor az üzenet az átmenet/üzenetküldés/művelet feltétele.
    - „Teljes” (kitöltött) nyílhegy: szinkron üzenetküldést, a művelet/átmenet „hívását” jelöli.

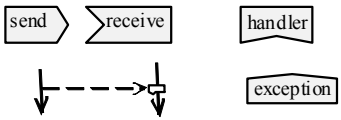
Üzenetküldés esetén az üzenet nevének megadása azt jelöli, hogy a célelemen belül definiált műveletet hívódik meg. Az átmenet vagy üzenetküldés esetén jelölhető a többszörösség, illetve az opcionalitás.

Általában véve: az irányítatlan szaggatott vonal – a „csatolás” – az elemhez kapcsolódó további elemet jelöli.

A jelölések lehetővé teszik UML „használati eset diagramok” megadását.

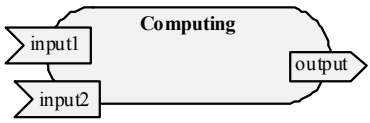
- Az UML „használati esetek” valójában a rendszer kívülről elérhető műveletei, melyeknél eltekintünk az időtartamuktól, így a használati esetek a strukturális elemként felvett rendszer műveleteiként ábrázolhatók.
- Azonos módon jelölhetők (pl. Web-) szolgáltatások, azok műveletei, illetve rendszerek/szolgáltatások kapcsolatai is.
- Aktor *használati esetei* az aktorból kiinduló, a műveletbe vezető szaggatott vonalú nyíllal jelölhetők. Részletesebb diagram esetén a vonalon jelölhető a késleltetés (hullám).
- UML „*include*”: az alapesetből kiinduló szaggatott vonalú teljes-nyílhegy.
- UML „*extend*”: itt (UML-hez képest fordítva:) az alapesetből indul ki szaggatott vonalú (normál) nyíl és jelölhető az opcionalitás.

Az eseményküldés külön alakzattal is jelölhető, amelyhez a küldött adat is hozzákapcsolható. Esemény fogadása megadható vagy egy egyszerű műveletként (akcióként) vagy egy külön alakzattal. Ez utóbbi azonos jelentésű egy szinkronizációs hívással.



69. ábra: Eseménygenerálás/ fogadás, ill. kivételkezelés jelölései

A küldés/fogadás jelei alkalmasak összetett számítás be- és kimenetének a megadására is (ill. a művelethívás küldött/fogadott paramétereinek megadására is).



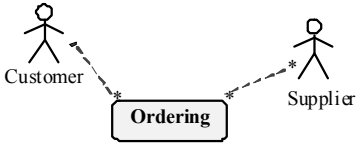
70. ábra: Számítás

A jelölésekkel általános rendszerek leírhatók (mind üzleti, mind informatikai vagy egyéb rendszerek). Befejezésként nézzünk a jelölésekre néhány példát.

Példánkban az *Ügyfelek* és *Beszállítók* kapcsolatban vannak egymással:

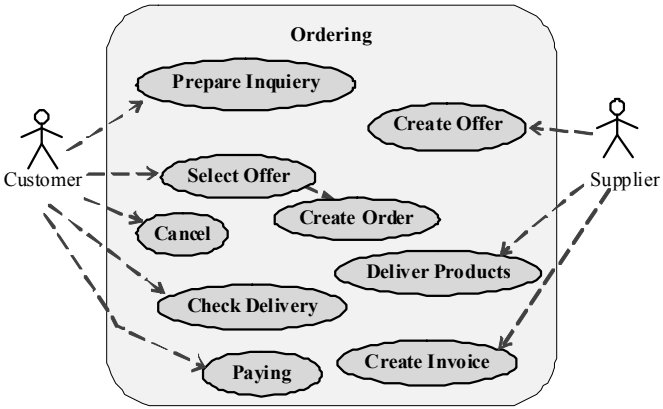
71. ábra: Ügyfelek és Beszállítók kapcsolata

Ez a kapcsolat a *Megrendelés* tevékenység („együttműködés”) folyamatán keresztül történik:



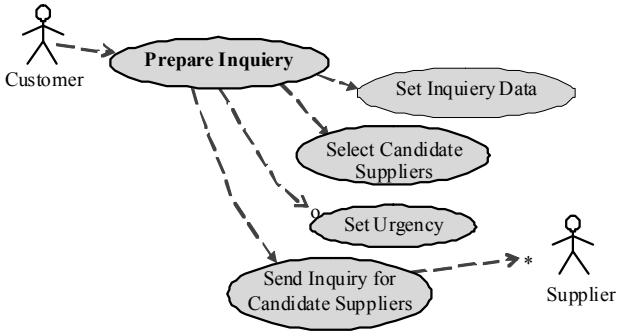
72. ábra: A Megrendelés közös tevékenység

Megadhatók az együttműködés műveletei („használati esetek” – „funkcionalitás”):



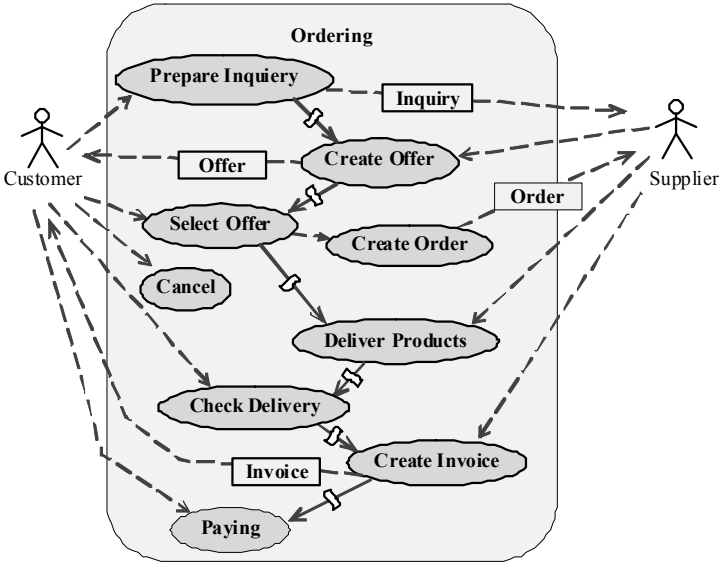
73. ábra: Műveletek („funkcionalitás”)

Az egyes funkciók tovább részletezhetők (UML „*include*” és „*extend*”):



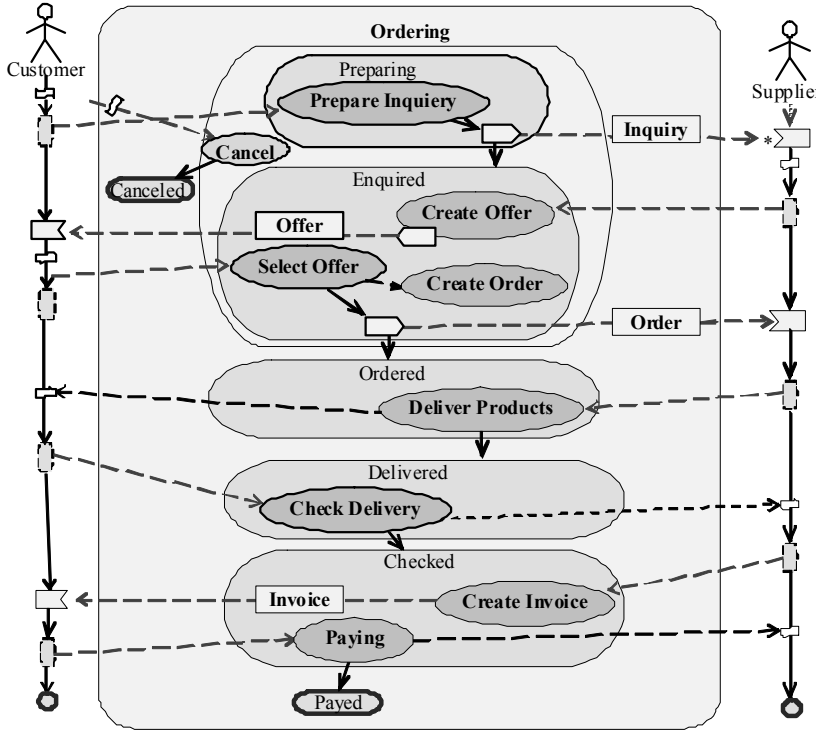
74. ábra: Művelet részletezése

Megadható az időbeliség és a küldött/fogadott adatok:



75. ábra: Műveletek („funkcionalitás”)

A modell tovább pontosítható az állapotok (ill. az azokban végrehajtható műveletek) megadásával. Ábrázolható a résztvevők (participants) tevékenységsorozata („életvonala”), illetve az általuk opcionálisan vagy kötelezően végrehajtandó feladatok („task”) sorrendisége.



76. ábra: Műveletek („funkcionalitás”)

A modellnek megadható egy strukturált szövegű (pl. XML) változata, illetve ugyanez megadható egy speciális, angolhoz hasonlító nyelven is. Ez a kísérleti nyelv jelenleg 15 mondat/almondat ill 11 „operandus” formát használ, pl: *„a/an”*: formális paraméter, *„the”*: aktuális paraméter, ~ : az angol „it” (C++ *this*) rövidítése, stb.

A szerkezetekkel megadhatók a strukturális viszonyok, állapotok, állapotváltások, műveletek, a résztvevők, a végrehajtható/végrehajtandó feladatok, vezérlési szerkezetek, stb. A megfelelő program („engine”) képes végrehajtani és nyomon követni a megadott folyamatot. A példánk vázlata pl. lehet a következő:

```
an ordering of a customer:
  ~ may have an inquiery
  ~ may have candidate supplier participants
  ~ may have offers
  ~ may have a selected offer
  ~ may have a supplier participant what is the supplier of the selected offer
  ~ may have an order

  ~ can be candidate
    the customer can cancel the ordering
      ~ is canceled
    preparing
      create an inquiry of the ordering
      the customer can set the inquiery data
      the customer can add candidate suppliers
      the customer can remove candidate suppliers
      when the customer finish preparing
        ~ is enquired
    ~ can be enquired
      firstly
        for each cadidate supplier of the inquery
          send the inquiery for the candidate supplier
          the candidate supplier may create an offer
      when an offer is received from a supplier
        it is an offer of the ordering
      when the customer creates an order of an offer
        the selected offer of the ordering is the offer
        send the order for the supplier
        ~ is ordered
  ~ can be ordered
    the supplier must deliver
    the customer must receive the delivery
      ~ is delivered
  ~ can be delivered
    the customer must check the delivery
      ~ is checked
  ~ can be checked
    the supplier must create an invoice
    the customer must pay
      ~ is  payed
  ~ can be payed
  ~ can be canceled
```

Néhány részlet az átalakított XML-formából:

```
<entity name="ordering" of="customer">
  <has name="inquiery" mandatory="false"/>
  <has name="candidate supplier participant" multiple="true"/>
  <has name="offer" mandatory="false" multiple="true"/>
  <has name="inquiery" mandatory="false" />
  <has name="selected offer" mandatory="false">
  <has name="supplier participant" value="selected offer. supplier"/>
  <has name="order" mandatory="false">

  <state name="candidate">
    <task name="cancel" mandatory="false">
      <when name="cancel">
        <set name="cancel"/>
      </when>
    </task>
  </state>
...
  <state name="ordered">
    <task for="supplier" name="deliver" mandatory="true"/>
      <when name="delivered">
        <set name="delivered"/>
      </when>
    </task>
  </state>
  <state name="delivered">
    <task for="customer" name="check the delivery" mandatory="true"/>
      <when name="OK">
        <set name="checked"/>
      </when>
      ...
    </task>
  </state>
  <state name="checked">
    <task for="supplier" name="create invoice" mandatory="true"/>
      <when name="OK"><where name="invoice"/>
        <send name="invoice" to="customer"/>
      </when>
    </task>
    <task for="customer" name="pay" mandatory="true"/>
      <when name="OK">
        <set name="payed"/>
      </when>
    </task>
  </state>
  <state name="payed"/>
  <state name="canceled"/>
</entity>
```

Informatics and especially the development of information systems may be thought of as a catalyzer that, through its problems and toolsets, helps in the construction of a sound ideology for the representation and acquisition of information. The goal is to briefly define a theoretical approach, which once could be the basis of constructing an exact conceptual framework for software development, as well as of establishing the most fundamental concepts of generic information acquisition and representation, all with mathematical accuracy.

Az informatikát, azon belül is az információs rendszerek fejlesztését tekinthetjük egy olyan katalizátornak, amely problémáival, illetve eszközkészletével segít kialakítani egy tiszta fogalomrendszert az információ ábrázolásáról, illetve az ismeretszerzésrõl. A cél egy olyan elvi megközelítés felvázolása, amely egyszer elvezethet a rendszerfejlesztés egzakt fogalmi kereteinek kialakításához, valamint az általános értelemben vett ismeretszerzés és reprezentáció legalapvetõbb fogalmainak matematikai precizitású tisztázásához.