A PROCESS ORIENTED APPROACH FOR QUALITATIVE MODELING AND ANALYSIS OF DYNAMICAL SYSTEMS

Agnes Janson-Fleischmann and Gerhard Sutschet
FhG - IITB, Sebastian-Kneipp-Str. 12/14, 7500 Karlsruhe 1, FRG

Abstract: The representation of dynamical systems according to the principles of qualitative modeling and analysis has two main aspects: The structural model representing the structure of real world objects in the system, and the description of the processes that run on the system. The second is content of this paper. We introduce a formal qualitative description language for processes based on QP-Theory of Forbus. Important extensions have been made in order to represent processes not just in a mainly continuous domain like physics. They concern the introduction of discrete processes, temporal relations between processes, and the description of processes at different levels of abstraction in form of a hierarchical process model. The possible behaviors of a system can be determined by qualitative reasoning about processes, which is performed in the behavior analysis.

1. Introduction

When building a deep model of a dynamical system, i.e. a changing physical, biological, chemical or economical system, we have to distinguish two parts:

- the description of the structure of real world objects, called the *structural model*. This model contains the description of all components of a system, their arrangement and properties;
- the description of the changes occurring in the structural model. These changes are modeled in processes. The set of all processes in a model form the *process model*.

We do not want to give the details for description of the structural model in this paper, but we present an interface from the process model to the structure model for referencing the necessary objects for the activity of a process. The structural model can be described in a component oriented description language, e.g. COMODEL [DiKi, Kipp], which is developed at our research institute.

CAPAS (CAlculus for Process Analysis and Simulation) is an approach for deep modeling and analysis of real processes using a formal, qualitative process description. The behavior of a dynamical system is described in a set of asynchronous processes with temporal restrictions that correspond through changes of parameters, existence or geometrical arrangement of objects. Up to now we have considered only changes of parameters. The modeling is characterized by the following features:

- Representation of causal relationships.
- Representation of temporal relationships:
  Not all dependencies between processes can be expressed as inherent dependencies such that one process activates another. We also need a temporal calculus for explicit representation of temporal relationships that are known at modeling time.
- Representation of continuous **and** discrete processes:
  Former approaches in Qualitative Physics have shown that continuous processes are an adequate entity for describing change in physical systems. Moreover there exist changes that are discrete in nature, e.g. in catastrophe theory, or that can be seen as discrete at a certain level of abstraction, e.g. the description of electronic circuits at level of digital logic. It is a non-trivial problem to handle both entities at the same time.
- Description of a hierarchical process model:
  The behavior of a system can be represented at different levels of abstraction.

After modeling a system, the *behavior analysis* can be performed. It results in a *behavior graph*, which describes all possible behaviors at a certain level of detail that is predefined by the user. In real systems the processes always start in an initial state and run in a special scope, i.e. relationships that hold in the initial state and that remain unchanged during the whole process. Therefore behavior analysis takes regard to a scope and initial state. Behavior analysis can be viewed as a translation of the process description to a state-transition diagram (the behavior graph) by analyzing the inherent and temporal dependencies between subprocesses.

The approach described in this paper is an extension of the QP-Theory of Forbus [For1]. The differences are that QP-Theory can not handle discrete changes, temporal relations between processes, and hierarchical process models. In opposition to the QP-Theory approach, Weld [Wel1,Wel2] sees the necessity for discrete processes. The main disadvantages of this approach for a general use are that there are no atomic continuous processes, that the automatically aggregated processes are usually not logical coherent, and that the user can not define aggregated processes as logical coherent sets of subprocesses.

With regard to qualitative analysis the work of De Kleer, Brown [DKBr], Kuipers [Kuip] and Voss [Voss] are also relevant. De Kleers IQ-Analysis is useful for describing the structural model and functional dependencies between parameters, but not for describing the changes themselves as separate entities. The HIQUAL

system of Voss allows structural modeling and modeling of causal and temporal relationships, but here also the description of change in separate units is not possible. Kuipers restricts himself to a direct qualitative interpretation of differential equations which are used in conventional simulation systems.

Chapter 2 defines processes as entities for modeling dynamical change. The syntax and informal semantics of a process formalism are defined and a detailed example is presented. Chapter 3 gives a method for predicting all possible behaviors of the system model by means of qualitative reasoning.

2. The Process Formalism

2.1 Definitions

In general a *process* is a change of a system over time. In the case of CAPAS we can define a process as a change of the structural model. In our first approach we concentrate to changes of parameter values.

The changes affected by a process are called *influences* of the process. A process always has a *status*; it is either *active* or *inactive*. Influences of a process operate if and only if the status of the process is active.

Parameters are to hold characteristic properties of objects (*object parameter)* or processes (*process parameter)*. From an epistomological point of view object parameters are to be defined in the structural model and process parameters in the process model. This is not essential for the operation of CAPAS but should be done in view of adequateness and explainability of the model.

One characterization of qualitative modeling is that parameters have *qualitative domains* consisting of symbolic values. A parameter is described by two values:

- the amount, called A-value, denoted by A(<parametername>) or in short <parametername>,
- the derivation, called D-value, denoted by D(<parametername>).

Changes of parameters may be described discretely or continuously. The domain of the A-values of a continuously changing parameter has to be at least partially ordered (we assume a total order), while the domain of a discrete changing parameter may be unordered. Furthermore in order to test the equality change law [For1] we have to distinguish points and intervals in the domain of a continuously changing parameter. The convention is that interval symbols are to be enclosed in parenthesis.

The D-values of parameters will always be interpreted in the ordered, qualitative domain [-, 0, +].

Allowing to specify temporal relations between processes explicitely we use the interval based logic introduced by Allen [All1,All2]. The 13 relations defined there are called *Allen relations.* The temporal relations between two processes J and K are denoted by

$J \{ar_1, ar_2, ..., ar_n\} K$, $ar_i \in$ Allen relations, with the semantics:

$\exists\ 1 \le i \le n: J\{ar_i\}K = tt \ \wedge\ \forall\ 1 \le j \le n, j \ne i: J\{ar_j\}K = ff.$

2.2 A Formalism for Describing Processes

In this section we will define a syntax for describing processes. We will introduce the semantics informally by explaining the meaning of the different slots. The semantics is given in chapter 3 by mapping the syntactic description into a set of state-transition diagrams.

2.2.1 A Syntax for Processes (in BNF)

```
<process> ::=
   process          <processname> ;
  [objects          <objectreference> { , <objectreference> }; ]
  [params           <parametername> : <domain>
                    { , <parametername> : <domain>} ; ] ②
  [subprocesses     [<processname> { , <processname>}] ;
                    [<processname> <AR> <processname>
                       { , <processname> <AR> <processname>}] ; ] ②
   conditions       [<log-expr> { , <log-expr> }] ; ①
  [relations        <log-expr> { , <log-expr>} ; ]
   influences       [<influence> { , <influence> }] ;

<objectreference>  ::= <object> | <constituent>

<object>           ::= is-of-type (<objectname>, <objecttypename>)

<constituent>      ::= constituent-of (<objectname>, <constituentname>,
                                       <objectname>) ②

<domain>           ::= '[' <value> { , <value>} ']' ③

<value>            ::= <valuename> | (<valuename>) ②

<AR>               ::= '{' <allenrelation> { , <allenrelation>} '}' ③

<allenrelation>    ::= m | mi | > | < | = | d | di | s | si | f | fi | o | oi
```

<log-expr> ::= <parameter> <rel> <parameter> |
<parameter> <rel> <constant> |
<constant> <rel> <parameter> |
(OR (<log-expr>) (<log-expr>) {(<log-expr>)}) |
(AND (<log-expr>) (<log-expr>) {(<log-expr>)}) |
(NOT (<log-expr>)) ④

<rel> ::= < | > | =

<influence> ::= <dir-cont-infl> | <indir-cont-infl> | <discr-infl>

<dir-cont-infl> ::= D(<parameter>) = + | D(<parameter>) = -

<indir-cont-infl> ::= <parameter> <prop> <parameter>

<prop> ::= >>+ | >> -

<discr-infl> ::= <parameter> = <constant>

Remarks:

① The condition T is assumed, if no entry exists.

② Non-terminal symbols with the extension 'name' are identifiers defined in the usual way (e.g. like in PASCAL [JeWi]).

③ The characters in quotes '{', '}', '[', and ']' respond to the notation of processes, i.e. they are no meta constructors of the BNF.

④ In the example later on we also use arithmetic operands within logical expressions. Strictly speaking we have to define the arithmetic operation extensional in case of symbolic domains or have to introduce a general interval arithmetic based on qualitative domains with real values as points and interval ranges. These extensions have been developed but are not treated in this paper.

2.2.2 The Meaning of the slots

process : The name of the process is specified. For further explanations this process is called P.

objects : Connection to the structural model. References to all objects, which are necessary for P to run. The parameters of the object defined in the structural model get inherited to P.

params : Definition of process parameters together with their qualitative domain. Overdefinition of the domains of object parameters.

subprocesses : Definition of the names and temporal relations of subprocesses. The declaration of the temporal relations is optional. They constrain the number of possible behaviors of the system and make the analysis of behavior (chapter 3) more efficient.

conditions : Activity conditions for process P.
There are two necessary (and also sufficient) conditions to be satisfied for a process being in status active during time-interval J:

1. The conjunction of *all* activity conditions evaluates 'true'.
2. The status active is possible w.r.t. the temporal relations declared in the subprocesses-slot of the father process.

relations : Relations that hold during the activity of P. They act as additional constraints on the possible values of parameters. They are not used for automatic derivation of further influences on parameters, like in De Kleer, Brown [DKBr] where confluences are derived from equations.

influences : Changes of parameters induced by the process. The changes are described either discrete (X = b, where X is a parameter and b a constant) or continuous. For continuous changes we distinguish *direct influences* of a process to a parameter X (D(X) = + or D(X) = -) and *indirect influences* from one parameter X to another Y (X >>+ Y or X >>- Y).

2.3 Example: Preparing Espresso

This apparatus for preparing espresso may be structurally modeled like in Fig. 2.3.

To explain the operation of the system we will start from the following state:

1. The cooker is turned off, the heat.source is cold.
2. The espresso.machine is filled, i.e.:
   - there is water in the water.pot,
   - the pressure in the water.pot is low,
   - the coffee.pot is empty, and
   - the powder container is filled up with coffee powder.
3. The espresso.machine is on the cooker.

When switching on, the heat.source will become hot. A heat.flow from heat.source to water.pot is started. Heating up the water will effect the generation of steam.

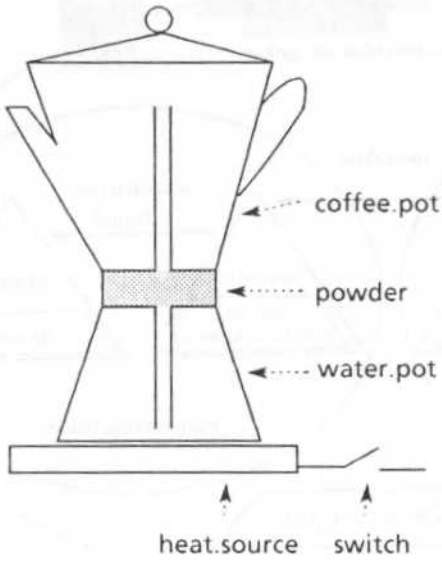


Fig. 2.2: Espresso Machine

This increases the pressure in the water.pot, what in turn will cause the hot water to be pressed through the stand-pipe and coffee powder into the coffee.pot. The process ends when neither the water.pot nor the stand-pipe contain any liquid.

With regard to the limited extent of this paper we will confine us to one aggregated process preparing-espresso with the following four subprocesses:

1. switching-on and switching-off:
   We assume an ideal heat.source, i.e. after switching on the heat.source is immediately hot, in the moment of switching off it is immediately cold.
2. boiling-water:
   This process models the generation of steam and pressure in the water.pot.
3. transporting-liquid:
   Transport of liquid from water.pot to coffee.pot.

We do not care about the process of heating the surroundings of the system, the chemical process of transmuting the water into espresso, and some other processes one can imagine.

We give a pictural instead of a verbal description of the structural model in Fig. 2.3., because we have not introduced the description language for objects. Therefor the domains of the object parameters are defined under the slot *params* in the process description.

process preparing-espresso;

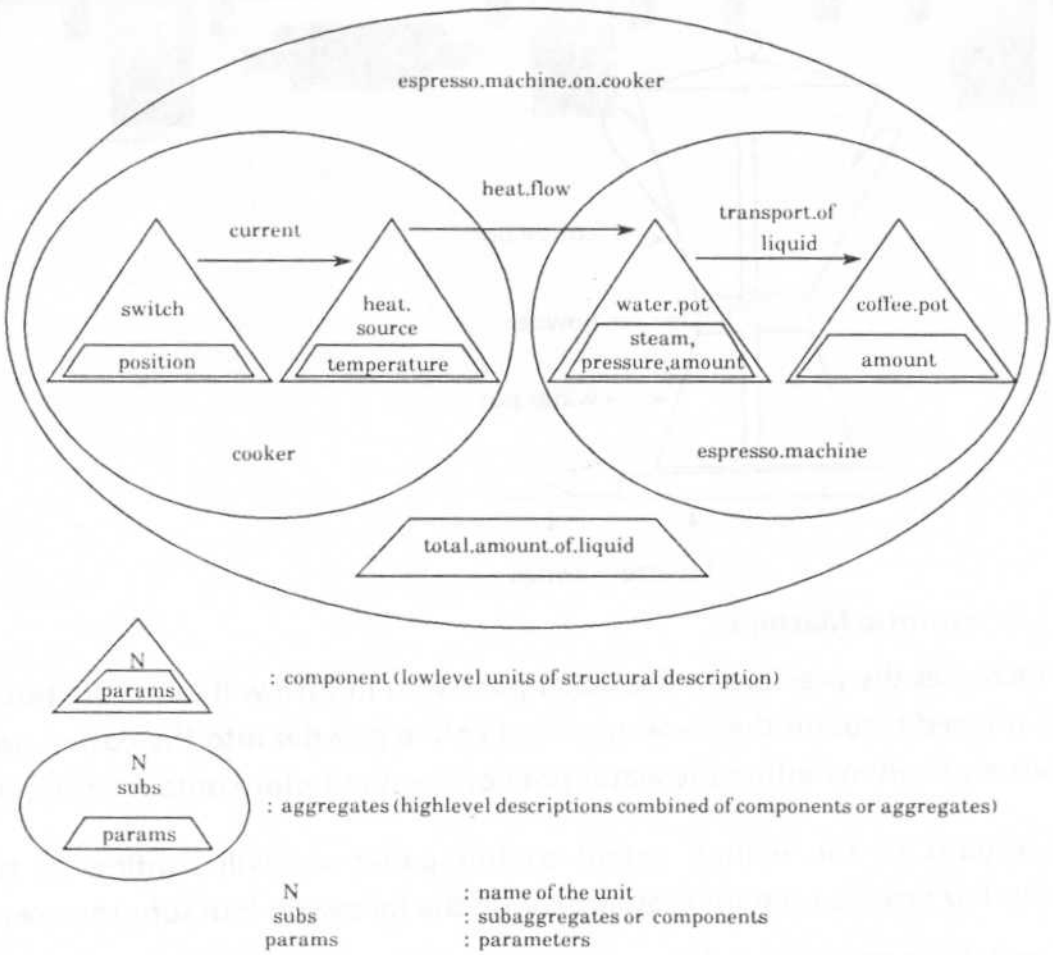


Fig. 2.3: Structural Model of the System

objects *is-of-type* (?espr.mach.on.cook, espresso.machine.on.cooker),
constituent-of (?cooker, cooker, ?espr.mach.on.cook),
constituent-of (?espresso.machine, espresso.machine,
?espr.mach.on.cook),
constituent-of (?switch, switch, ?cooker),
constituent-of (?heat.source, heat.source, ?cooker),
constituent-of (?water.pot, water.pot, ?espresso.machine),
constituent-of (?coffee.pot, coffee.pot, ?espresso.machine);

params position(?switch) : [on, off],
temp(?heat.source) : [cold, hot],
total.amount.of.liquid(?espr.mach.on.cook) :
[0, (0-total.amount), total.amount,
(total.amount-max.amount1), max.amount1],
amount(?water.pot) :
[0, (0-total.amount), total.amount,
(total.amount-max.amount2), max.amount2],

```
                amount(?coffee.pot) :
                    [0, (0-total.amount), total.amount,
                     (total.amount-max.amount3), max.amount3],
                steam(?water.pot) :  [0, greater.than0],
                pressure(?water.pot) : [low,  (low-rising.pressure),  rising.pressure,
                                        (rising.pressure-high), high];
  subprocesses  switching-on, switching-off, boiling-water, transporting-liquid;
                switching-on {m} boiling-water,
                switching-on {<} transporting-liquid,
                transporting-liquid {m,<} switching-off,
                boiling-water {m,<} switching-off;
  conditions    total.amount.of.liquid > 0;
  relations     amount(?water.pot) + amount(?coffee.pot) =
                    total.amount.of.liquid;
  influences    D(amount(?water.pot)) = -,
                amount(?water.pot) >>- amount(?coffee.pot);

process switching-on;
  conditions    position(?switch) = off;
  influences    position(?switch) = on;
                temp(?heat.source) = hot;

process switching-off;
  conditions    position(?switch) = on;
  influences    position(?switch) = off;
                temp(?heat.source) = cold;

process boiling-water;
  conditions    temp(?heat.source) = hot,
                amount(?water.pot) > 0;
  influences    D(steam(?water.pot)) = +,
                steam(?water.pot) >>+ pressure(?water.pot);

process transporting-liquid;
  conditions    (OR (pressure(?water.pot) > rising.pressure)
                    (pressure(?water.pot) = rising.pressure)),
                amount(?water.pot) > 0;
  influences    D(amount(?water.pot)) = -,
                amount(?water.pot) >>+ pressure(?water.pot),
                amount(?water.pot) >>- amount(?coffee.pot);
```

The specified temporal relations between the subprocesses are demonstrated in Fig. 2.4.

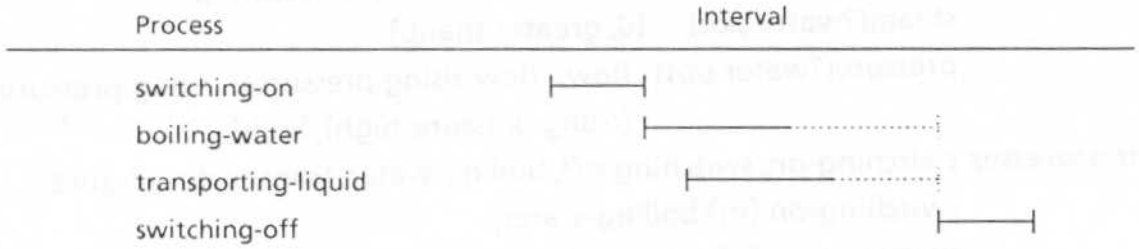


Fig. 2.4: Temporal Behavior of the Subprocesses

3. Analysis of Behavior

3.1 Temporal and Inherent Dependencies between Processes

Section 2.2 presented a formalism for process modeling, which allows to describe the behavior of a dynamical system in distinct levels of abstraction (or inverse levels of detail) by a hierarchy of processes, see Fig. 3.1. The subprocesses P_1, ..., P_n of the

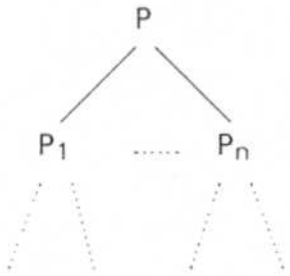


Fig. 3.1: Hierarchy of Processes

aggregated (root-)process P all together describe the behavior of P in a higher level of detail, where each P_i describes a partial aspect of P. The same mechanism is valid for all processes in the hierarchy and their direct or indirect subprocesses. For performing the behavior analysis, the level of detail in the hierarchy must be defined. In words of graph theory this means, that a *cut* through the tree must be specified. Informally, a cut is a subset of nodes such that it contains exactly one node of every path from the root to a leaf node. So, for example, the root is a trivial cut and all nodes surrounded by the closed line of Fig 3.2 form a cut. Every cut describes the complete behavior of the system at a fixed level of detail. The set of all processes in the cut is called *focus*. It is the input for behavior analysis.

A number of precomputations concerning the scope of slots of predecessing processes are to be made before starting the actual behavior analysis. Only one of

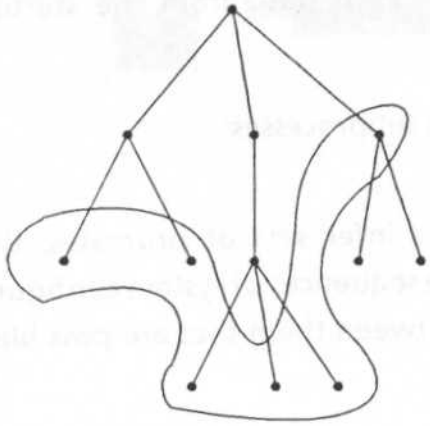


Fig. 3.2: Hierarchy of Processes with a Focus

them is discussed here: The computation of temporal relations between processes in a cut, which are not brothers. In CAPAS only the temporal relations between brothers may be known and can be fetched from the fatherprocess. But not all processes in a cut are brothers. For two non-brothers the temporal relations can be determined in a way described in [All1]. It is based on the implicit assumption, that by default all subprocesses are {s,d,f} to their fatherprocess. So, by transitivity, the temporal relations between each pair of processes in the hierarchy can be computed. See for example Fig. 3.3 with the assigned time diagram. The cut consists of P_2, P_3, P_4. By transitivity we get P_3 {<} P_2 and P_4 {<} P_2.

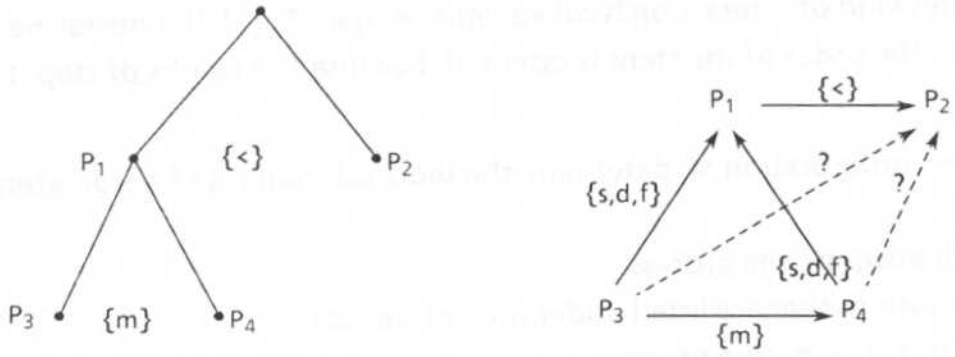


Fig 3.3:Example for Determining Temporal Relations in a Cut of a Process Hierarchy

The goal of the *behavior analysis* is to analyse the dependencies between the subprocesses in a cut $\{P_1,...,P_m\}$ starting from an initial state. The analysis results in a *behavior graph* that describes all possible behaviors at level of detail specified by the cut . Nodes of the behavior graph are system configurations (in short: configurations). A configuration is a set of processes, which are active in the same time-interval (subsets of $\{P_1,...,P_m\}$). The graph is connected by directed edges, which are interpreted as *possible transitions between configurations.* They describe the activation and deactivation of processes. The *startnode* is a

configuration in which no process is active. Each path from the startnode to another node describes one possible behavior.

There are two kinds of dependencies between subprocesses:

1. Temporal dependencies:
   By temporal reasoning it is possible to infer sets of processes, that can simultaneously be active and the possible sequences of system configurations, i.e. the configurations and transitions between them that are possible under the temporal restrictions.

   Example:

   Let P_1 and P_2 be the only subprocesses of P, and P_1 {o} P_2 a given temporal relation. The behavior of P can be described by

   $\{P_1\} \rightarrow \{P_1,P_2\} \rightarrow \{P_2\}$

   i.e. at first P_1 is active, then both subprocesses are active, then P_2.

2. Inherent dependencies:
   Processes interact by shared influences along common parameters. Different active processes can affect the same parameter directly or indirectly or a process can affect a parameter mentioned in the activity condition of another process: the condition of an inactive process can become valid or the condition of an active process can become invalid (change of status). The analysis of this kind of interaction is called *limit analysis* [For1]. It is performed in two steps. The order of the steps is essential, because the results of step I) enter step II).

   I) For system configuration SC determine the induced changes of parameters as follows:

   Direct influences of *one* process:

   All expressions that are listed under slot 'influences';

   Indirect influences of *one* process:

   Only possible for continuously influenced parameters;

   $Q_1 >\!\!>+\ Q_2 : D(Q_1) = q \Rightarrow D(Q_2) = q$

   $Q_1 >\!\!>- \ Q_2 : D(Q_1) = q \Rightarrow D(Q_2) = \overline{q}$

   where $q \in \{-,0,+\}$, $\overline{q} = \begin{cases} + & \text{if } q = - \\ 0 & \text{if } q = 0 \\ - & \text{if } q = + \end{cases}$

   Multiple influences on the same parameter X are combined as follows:

   Let $D_1(X)$ and $D_2(X)$ be single influences or intermediate results. The table specifies their concatenation $D_1(X) \circ D_2(X)$:

$D_2(X)$ \ $D_1(X)$	-	0	+
-	-	-	-,0,+
0	-	0	+
+	-,0,+	+	+

The table shows the non-determinism caused by contrary influences. Sometimes a conflict may be resolved by the previously known temporal relations. It may also be possible (but is not done in this approach) to make a refined subdivision of the domain of the D-values. Nevertheless, a conflict resolution is not always possible.

Example:

Assume the process definition of section 2.3.

Let SC_1 = {boiling-water, transporting-liquid} be a configuration.

The following changes of parameters are caused in SC_1:

D(amount(?water.pot)) = -, D(amount(?coffee.pot)) = +,

D(steam(?water.pot)) = +,

D(pressure(?water.pot)) $\in$ {-,0,+}, because boiling-water has positive, transporting-liquid negative influence on pressure.

Let SC_2 = {switching-on} be a system configuration consisting of a discrete process.

Changes of parameters:

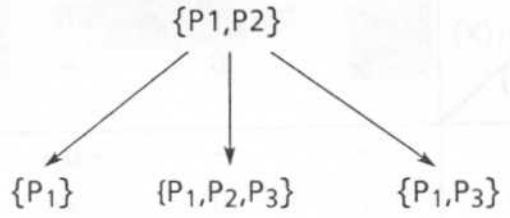
A(position(?switch)) = on, A(temp(?heat.source)) = hot.

II) For a system configuration SC determine, which processes can change their status due to the parameter changes determined in I). If more than one process can change its status, it is not realistic to assume that they all do it in the same moment. Instead, all temporal sequences in change of status are to be considered. By this you get all successing configurations $SC_1,...,SC_n$, which are possible under inherent dependencies.

Example:

Let $\{P_1,P_2\}$ be a system configuration. Assume that the result of the limit analysis is that P_2 becomes inactive and P_3 active. The following transitions are possible:

Here an essential advantage of the use of previously known temporal relations can be seen: it is possible that some transitions can bee excluded because they are not valid under temporal restrictions.



The analysis of behavior is based on two fundamental assumptions:

- all changes are caused directly or indirectly by processes;
- beyond the influences induced by the processes defining a system, there are no further influences ('closed world assumption').

These assumptions imply that the D-value of all non-influenced parameters is 0.

The analysis of the temporal and inherent dependencies are essential steps in generating the behavior graph. For a system configuration SC, the allowed successor configurations SSC are determined as follows:

1. TSC = successor configurations, that are possible under temporal dependencies
2. ISC = successor configurations, that are possible under inherent dependencies
3. SSC = TSC ∩ ISC .

This is illustrated in the next section by means of the espresso-example.

3.2 Example: Preparing Espresso

For the process preparing-espresso, its subprocesses, and the initial state, which has been specified in section 2.3, we get the behavior graph of Fig 3.4.

As an example, the determination of the successing configurations of * and ** is discussed in detail:

Successing configurations of *: SC = {boiling-water}

A-values known: position(?switch) = on, temp(?heat.source) = hot,
water-pot. amount > 0

Changes in parameters: D(steam(?water.pot)) = +, D(pressure(?water.pot)) = +

TSC = {{boiling-water, transporting-liquid}, {transporting-liquid}}

ISC = {{boiling-water, transporting-liquid}, {boiling-water, switching-off},
{switching-off, boiling-water, transporting-liquid}}

SSC = {{boiling-water, transporting-liquid}}.

Successing configurations of **: SC = {boiling-water, transporting-liquid}

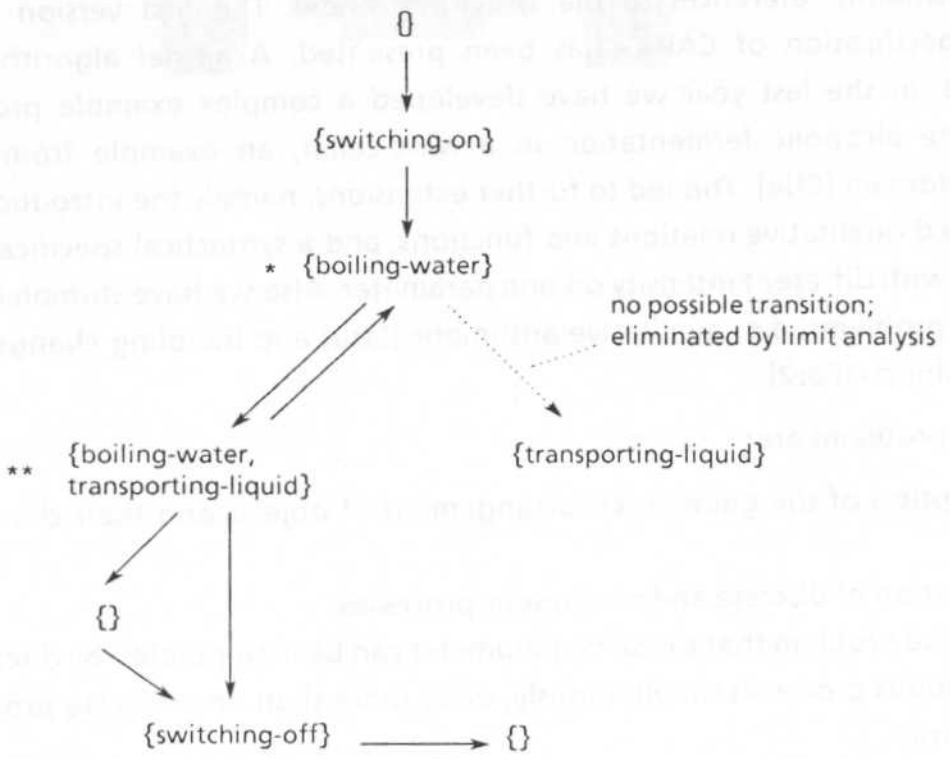


Fig 3.4: Behavior Graph for Process Preparing-Espresso

A-values known: position(?switch) = on, temp(?heat.source) = hot,
amount(?water.pot) > 0

Changes in parameters:

D(amount(?water.pot)) = -, D(amount(?coffee.pot)) = +,
D(steam(?water.pot)) = +, D(pressure(?water.pot)) $\in$ {+,0,-}

Possible changes in status:

switching-off can become active,
transporting-liquid can become inactive, because pressure is decreasing,
transporting-liquid and boiling-water can become inactive simultaneously, because amount(?water.pot) becomes 0.

TSC = {{boiling-water}, {transporting-liquid}, {}, {switching-off}}

ISC = {{boiling-water}, {}, {switching-off}, {switching-off, boiling-water}, {switching-off, boiling-water, transporting-liquid}}

SSC = {{boiling-water}, {}, {switching-off}}

4. Conclusion and Open Problems

In this paper an extended calculus for qualitative modeling and analysis of dynamical systems has been introduced. It allows to represent the behavior of

processes, including references to the structural model. The first version of a syntactical specification of CAPAS has been presented. A kernel algorithm is implemented. In the last year we have developed a complex example process model for the alcoholic fermentation in a wine-cellar, an example from the biochemical domain [BIJa]. This led to further extensions, namely the introduction of user-defined qualitative relations and functions, and a syntactical specification of influences with different intensity on one parameter. Also we have stumbled on some known problems, e.g. qualitative arithmetic [Stru] and handling changes in existence of objects [For2].

Further open problems are:

- The description of the geometrical arrangement of objects and their change-ment
- The integration of discrete and continuous processes:
  It leads to the problem that the same parameter can be manipulated by discrete and continuous processes simultaniously, or by more than one discrete process simultaniously.
- Focussing in more general process structures:
  In this paper we assumed the process hierarchy to be a simple tree, where the focus was defined to be a cut through the tree. Unfortunately case studies, e.g [BIJa], have shown, that the structure of a simple tree is too specialized. We need more general structures such as AND/OR trees or trees with typed edges etc. The problem is to define the focus in the more general structure analogic to the cut in the simple hierarchy. For a detailled discussion see [Suts].
- Cyclic behavior
  Allens approach is unable to deal with cyclic and reentrant time structures. We need an extended time calculus for describing and handling cyclic behavior.

Acknowledgments

This work is sponsored by the BMFT (Ministery of research and technology of the Federal Republic of Germany). It is done in the project called TEX-B (Technical EXpert systems Basis).

References

[All1] J.F. Allen, Maintaining Knowledge about Temporal Intervals, Communications of the ACM 26 (11), 1983, 832-843.

[All2] J.F. Allen, Towards a General Theory of Action and Time, Artificial Intelligence 23 1984, 123-154.

[BlJa] M. Bläsius / A. Janson-Fleischmann, Qualitative Modellierung der Weinbereitung mit CAPAS, TEX-B Memo 18-87, FhG-IITB Karlsruhe, FRG, 1987.

[DKBr] J. De Kleer / J.S. Brown, A Qualitative Physics Based on Confluences, Artificial Intelligence 24, 1984, 7-83.

[DiKi] W. Dilger / J. Kippe, COMODEL : A Language for the Representation of Technical Knowledge, Proceedings of the 9th IJCAI, Los Angeles 1985, 353-358.

[For1] K.D. Forbus, Qualitative Process Theory, Artificial Intelligence 24, 1984, 85-168.

[For2] K.D. Forbus, The Problem of Existence, Proceedings of the Cognitive Science Society, 1985.

[JeWi] K. Jensen / N. Wirth, Pascal User Manual and Report, Springer-Verlag, 1974.

[Kipp] J. Kippe, COMODEL : Ein Repräsentationsformalismus für technische Expertensysteme. GWAI-86 and 2. Austrian AI-Conference, Ottenstein/Austria, 1986, 349-360.

[Kuip] B. Kuipers, Qualitative Simulation, Artificial Intelligence 29, 1986, 289-338.

[Stru] P. Struss, Mathematical Aspects of Qualitative Reasoning, Früchtenicht(ed), Technische Expertensysteme: Wissensrepräsentation und Schlußfolgerungsverfahren, Oldenbourg Verlag, 1987.

[Suts] G. Sutschet, Putting Structure on the Process Models in Qualitative Physics, Proceedings of the 12th IMACS World Congress '88 on Scientific Computation, 1988.

[Voss] H. Voss, Representing and Analyzing Causal, Temporal, and Hierarchical Relations of Devices, Dissertation, Fachbereich Informatik, Universität Kaiserslautern, 1986.

[Wel1] D.S. Weld, Combining Discrete and Continuous Process Models, Proceedings of the 9th IJCAI, Los Angeles 1985, 140-143.

[Wel2] D.S. Weld, The Use of Aggregation in Causal Simulation, Artificial Intelligence 30, 1986, 1-34.