S.Bandini, M.Bruschi, M.G.Filippini, A.Molesini

Dipartimento di Scienze dell'Informazione
Università di Milano
via Moretto da Brescia, 9
20134 Milano, ITALY
tel. +39 (2) 2772233

A logic programming approach to Qualitative Process Theory

ABSTRACT

In this paper a computational model of Qualitative Process Theory [6] based on logic programming is presented.
The model is based on a Qualitative Process Language (QPL). which is used to describe physical systems. A QPL interpreter implemented in Prolog (IQPL) allows inferences on such descriptions for reasoning about physical systems and domains.

1. INTRODUCTION

The high level of interest in the development of qualitative models of physical systems in A.I. [11] is evidenced by the remarkable number of researches under development in this subfield.
The use of logical languages to obtain commonsense and qualitative reasoning computational models of physical systems has also been subject of research: examples of the PROLOG language in the development of such models can be found in [12] [19].
This paper describes research work in the area of Prolog applications in Naive Physics: its main aim is the presentation of a computational model of Qualitative Process Theory [6] based on logic programming.
The reasons for this work are:

- the need of studying qualitative modeling of physical systems;
- the attempt to understand if and how logic programming could be useful to qualitative physics [7][8].

In particular, we chose QPT for the following reasons:

- analysing and modeling physical systems often involve reasoning in terms of changes and physical processes with incomplete informations;
- it is desiderable when reasoning on complex physical systems that the knowledge embedded in the models is incrementally augmentable.

A remarkable example of the use of Prolog in Qualitative Physics is the work of I.Bratko [3]: he developed a qualitative calculus and used it to model the electrical activity of the heart.
This work highlights the peculiarities of Prolog with relation to the logical aspects of QPT and presents a new interpretation, favoured by Prolog itself, of the inferential constructs of the theory.

The attention payed to the use of Prolog's inferential mechanisms and the adoption of technique of meta-interpretation [14][17] have allowed an almost integral realization of the QPT language.

In the following paragraphs we present a QPT computational model based on a language (QPL) for the description of physical domains and situations and an interpreter for QPL (IQPL) - implemented in Prolog - which provides the main primitives for reasoning about the descriptions of physical systems and domains expressed in QPL.

2. THE QPT COMPUTATIONAL MODEL

Qualitative Process Theory (QPT) [6] is a formal instrument for representing knowledge and supports commonsense reasoning about the dynamics of physical situations (Qualitative Dynamics).
QPT defines a primitive notion of physical process, which can be used to model theories of dynamics, intuitively understood as something which causes a change to objects over time.
Forbus proposes a computational model for QPT which makes use of a formal representation **language** to express properties and relations among defined objects in a domain.
To estabilish properties of objects such as existence, activability or influences, a set of basic **deductions** is defined which constitutes the inferential part of QPT.

Fig. 2.1. An example of definition of a physical process
A liquid flow occurs between two contained liquids if they have different pressures and there is an aligned path between them through which the liquid can flow.

Process Liquid-Flow
  Individuals:
    src a Piece_of_Stuff, Contained_Liquid(src),
    dst a Piece_of_Stuff, Contained_Liquid(dst),
    path a Fluid-Path, Fluid-Connected(path,src,dst)
  Preconditions:
    Flow-Aligned(path)
  QuantityConditions:
    A[pressure(src)] > A[pressure(dst)]
  Relations:
    Let flow-rate be a quantity
    A[flow-rate] > ZERO
    flow-rate α_{Q+} (pressure(src) - pressure(dst))
  Influences:
    I- (amount-of(src), A[flow-rate])
    I+ (amount-of(dst), A[flow-rate])

2.1 The QPT language

The QPT language is a formal tool for representing information on physical domains, objects and systems.
An example domain can be provided by liquids, in which objects can be 'pieces of liquids' [9] or containers and pipes. These can be combined together in a physical system.
A useful representation of a domain is given by the following syntax types of constructs:

- individuals or entities (for physical objects)
- relations among objects
- processes (for physical phenomena _ see fig. 2.1)

The description of a physical system at a given time is called the **scenario**: it specifies the collection of objects involved in the system, relations among them and inequality statements about their parameters.

2.2 The QPT deductions

In order to become a computational model, a representation for physical systems must support deductions [6].
The central assumption of the theory is that all changes in physical systems are caused, directly or indirectly, by processes (Sole Mechanism assumption). Behavioural analysis of system descriptions in a given physical situation can thus be performed, in an iterative way, as follows:

.. OBJECTS IDENTIFICATION:
all objects which may come into existence as a result of the activation of a process are determined by analysing all the process descriptions to match the ones mentioned in the *individuals* field of the descriptions to the ones currently available.

.. ACTIVE PROCESSES DETERMINATION:
all active processes are determined by evaluating the preconditions in the process descriptions and possibly by performing a search in some quantity space in order to deal with incomplete information;

.. PARAMETER CHANGES DETERMINATION:
all the changes in the system parameters are computed by considering all the influences exerted by the active processes on each parameter. Changes can influence the set of active processes which must then be recomputed according to a small set of intuitive rules.

3. FROM QPT TO QPL & IQPL

3.1 QPT revisited

A qualitative description of a physical system implicitly embeds commonsense knowledge on the behavior of the system itself, namely its **dynamic description.**
Two elements are needed to use the inferential apparatus of QPT to render this knowledge explicit:

- a **description**: commonsense knowledge on the domain to which the system belongs. Essential here is the description of the processes that can occur in such a domain. It is a **non-changeable** type of knowledge;

- a **situation**: *the instantaneous description of the system.* For example the initial state of a system described in a QPL scenario by a set of predicates represents a situation. This set of predicates has a **dynamical** nature.

The situation of a physical system is a set of logical predicates, which are considered simultaneously true.
The definition of *three fundamental operations* on such predicates allows the implementation of the basic deduction ability of our QPT computational model:

- **prove**: specifies *the way to conduct the proof of a predicate.* It requires the definition of a set of rules to establish whether a predicate *logically follows* from the predicates which describe the current situation. We say that a predicate P is *proved* in a situation S at time t if it can be deduced by the predicates which hold in situation S at time t, using r(P), the set of proof rules defined for P. That is

  proved(P,S,t) ≡ situation(S,t) $\Rightarrow_{r(P)}$ P

- **remember**: adds a predicate P - and any other predicate that a set of rules binds to P, identified with bound_to(P) - to the current situation.

  remember(P,S,t) ≡
  situation(S,t+1) = situation(S,t) U P U bound_to(P)

- **forget**: retracts a predicate P - and any other predicate bound to P - by the current situation.

  forget(P,S,t) ≡
  situation(S,t+1) = situation(S,t)\(P U bound_to(P))

Unlike K.D.Forbus, *we propose these primitives as the basic level of interaction with physical domain and system descriptions based on QPT*, whereas he addresses the basic deductions of the theory. Now they can be defined in terms of **prove**, **remember** and **forget**.

3.2 QPL: a realization of the QPT language for the represention of physical domains and systems

The availability of a formal language for descriptions is the starting-point for the development of a computational model for reasoning about physical systems.

This particular realization of QPT language, named QPL (Qualitative Process Language), provides:

i) A set of constructs to express the semantics of the relations which typically occur among objects and related parameters in a physical domain: **primitive phrases**.

ii) Structures which contain the above information for the description of physical objects and phenomena, named **parametrical descriptions**.
To these, we add further definitions such as scenarios.
Such parametrical descriptions can be collected in a library which can be used to represent properties of a physical domain.
Among them, we will refer to Individual Views - an abstraction of individuals whose existential state can vary over time - and to Processes - time dependent descriptions of events which cause changes - with the expression "Conditionalized Descriptions" (CD).
A set of CDs, called a vocabulary, is the most important object in a library constructed for the description of a physical domain.

The constructs which allow to make assertions using QPL in phrases are divided in different Syntax Types [15].
Each of the basic primitives must be defined for every syntax type: this specification defines the OPERATIONAL SEMANTICS of QPL.

The fundamental feature which marks our reinterpretation of the QPT language is its logical conception: QPL, in fact, is a language entirely based on predicate logic.

3.3 IQPL: an Interpreter for QPL

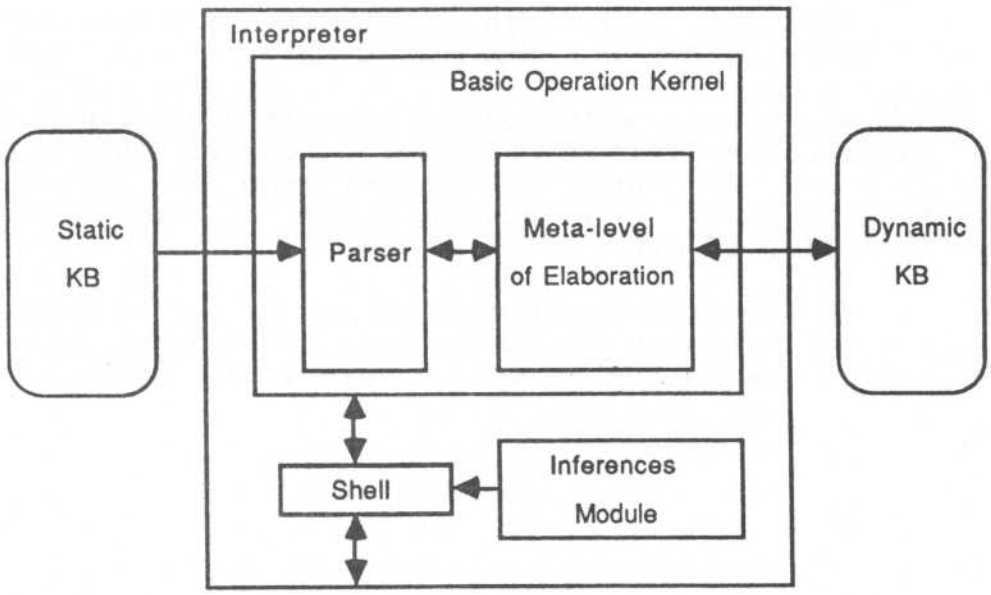
"Computing" the description of a physical domain in the QP Language means to carry out the basic inferences defined by QPT, i.e. respectively analysis of situation (elaboration), activity_determination, change_propagation, limit_analysis on the change of parameters.
Thus, a program devoted to qualitative reasoning based on the representation of a physical situation can be intuitively conceived as an "interpreter" of such a representation on which it must be able to operate suitable manipulations [4].
This review led to the definitions of an interpreter for QPL: it can be thought of as an abstract inferential mechanism for QPT, i.e. a tool which can automate the deduction activities needed to reason about physical systems.

Manipulations on sentences are defined by means of basic operations on the computational model. Thus we can see an interpreter for QPL as a *cluster* [6] which

- has *data structures*:
  - a ***static*** *knowledge base*, the so_called *description* (see 3.1);
  - a ***dynamic*** *knowledge base*, that is the *situation*;
- provides *primitives* for interaction with these knowledge bases (prove, remember, forget) which may be used in sophisticated reasoning activities.

The logical architecture of such an interpreter is sketched out in fig. 3.1.

Fig. 3.1.



This architecture is composed of four main modules:

- the **Meta-level of Elaboration** provides the three basic operations for QPL sentences. It is called a *meta-level* because it builds QPT inferences rules above Prolog's inferential mechanism.
  The predicate devoted to submit QPL phrases to operations is

```
kb Elaborate this Phrase
```

  where Elaborate can be instantiated to **prove**, **remember** or **forget**.
  For the first operation the specialistic rule is

```
kb prove the Type::Obj using KB_Form :-
  non(Obj) is_in_kb,!,fail ;
  KB_Form is_in_kb ;
  kb satisfy the Type::Obj ;
  Type::Obj has_true conditions ;
  query_the_user about Type::Obj.
```

  The operator 'about' allows the user to introduce knowledge that IQPL cannot deduce using its 'own' knowledge bases and rules;

- the **Parser** interfaces the static knowledge base with the Meta-level of Elaboration which requires information to process. The retrieval of the interested knowledge occurs specifying the parametrical description name and related field to reference, like in frame structures, as in the following query:

```
?- q_conditions of def(CatSynt) ::
     `Liquid_Flow`(c,d,pl) iS Quantity_Conditions.

CatSynt = process
Quantity_Conditions = a(pressure(c)) gt a(pressure(d))
```

  For each obtained phrase the *Syntax Type* is recognized. The rules of the Meta-level of Elaboration are parametrical with respect to the syntax types;

- the **Inference Module** contains the QPT basic deductions in terms of the operations provided by the Meta-level of Elaboration;

- the **Shell Module** accepts commands from a generic user, like requests of loading descriptions in Static KB or QPT deductions.

3.4 An example

Let's consider an example: the "communicating_vessels" system.

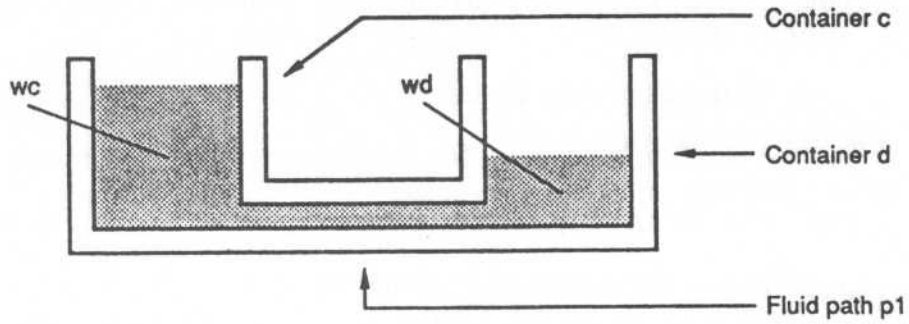


Fig. 3.2.

A QPL representation for such a system is the following:

```
def(scenario) :: vasi_comunicanti with
  (individuals: (c, d, pl) and
   facts: ('Container'(c), 'Container'(d),
           'Fluid_path'(pl)) and
   always: ('Fluid_connected'(inside(c),inside(d),pl),
            'Open'(c),'Open'(d),'Same_Shape'(c,d)) and
   individuals_in_situation: (c, d, pl) and
   facts_in_situation:
      (a(amount_of_in(c,water)) gt 'ZERO',
       a(amount_of_in(d,water)) gt 'ZERO',
       a(amount_of_in(c,water)) gt
              a(amount_of_in(d,water)))).
```

In the liquids domain, a fundamental process involved in descriptions of changing systems is 'Liquid_Flow', agreeing with the 'Contained_Stuff' ontology introduced by P.Hayes [9] and adopted by Forbus.
Such an approach allows macroscopic descriptions of physical phenomena which totally agree with the model proposed in the QPT. The following description of this process is the QPL form of the example of fig. 2.1 (for more details about the definitions of QPT objects using QPL see [1]):

```
def(process) :: 'Liquid_Flow'(Src,Dst,Path) with
  (individuals:
      ('Piece_of_Stuff'(Src),
       active('Contained_Liquid'(Src)),
       'Piece_of_Stuff'(Dst),
       active('Contained_Liquid'(Dst)),
```

```
        'Fluid_path'(Path),
        'Fluid_connected'(location(Src),
                          location(Dst),Path)) and
    preconditions: 'Aligned'(Path) and
    q_conditions:
        a(pressure(Src)) gt a(pressure(Dst)) and
    relations: (local(flow_rate,'Quantity'(flow_rate)),
                a(flow_rate) gt 'ZERO',
                flow_rate q_Dprop pressure(Src),
                flow_rate q_Iprop pressure(Dst)) and
    influences: ('I+'(amount_of(Dst),a(flow_rate)),
                 'I-'(amount_of(Src),a(flow_rate)))).
```

In the situation described in the "communicating_vessels" scenario, the only active Process Instance is 'Liquid_Flow'(wc,wd,pl), where wc and wd identify the entities of type 'Piece_of_Stuff' which correspond to the pieces of liquid contained in 'c' and 'd' respectively.
The activation of a Process - which occurs if its preconditions and q_conditions can be proved in the current situation - asserts the phrases contained in the 'relations' and 'influences' fields, that is, respectively, information about functional dipendencies introduced by the Process or about new entities related to it, and influences among quantities.

In this case, the Process is described by means of the 'flow_rate' of water in the fluid_path, which is proportional to the relative pressures of fluid in the containers. The sign of the influence exerted by the Process on the amounts of liquid in the containers allows to conclude that part of the fluid contained in c will be transferred in d, and that the Process will stop when the relative pressures become equal.

After elaboration of the scenario, IQPL "knows" the initial situation of the system. In this situation individuals that could logically exist as instances of the CDs will be searched for.
Such initialization of the dynamic knowledge base can be expressed using remember:

```
elaborate_scenario(S) :-
  remember all facts & always & facts_in_situation of S .
```

Besides the individuals present in the scenario, other entities _ introduced by the potentially active processes in the situation _ are retrieved by a deduction called 'elaboration'.
In this way all possible individuals deriving from an initial collection of objects are generated. Given a set of individuals and a vocabulary of CDs, the 'individuals' field in the vocabulary elements is used to find view and process instances which can potentially occur.

In our example the following istances are found:

```
Contained_Stuff(d,water,wd)
Contained_Stuff(c,water,wc)
Liquid(wd)
Liquid(wc)
Gas(wd)
Gas(wc)
Contained_Liquid(wd)
Contained_Liquid(wc)
Liquid_Flow(wd,wc,pl)
Liquid_Flow(wc,wd,pl)
```

The next step is to find Process and View Structure (collections of process and view instances which are active in a situation) through the inference 'determine_activity'. These structures represent what happens to individuals in the situation: they constitute a synthetic explanation of phenomena occurring in the situation and of objects involved. Follow the Prolog rules for this deduction...

```
det_activity :-
  instances is_worth CD_list,
  /* the set of CD instances obtained by elaboration */
  for_each(member(CD_istance,CD_list),
           activate(CD_instance)).

activate(CD_instance) :-
  prove all preconditions & q_conditions of
  CD_instance,
  remember this CD_instance.
```

... and the related results:

```
Process Structure:
   Liquid_Flow(wc,wd,pl)

View Structure:
   Contained_Stuff(d,water,wd)
   Contained_Stuff(c,water,wc)
   Liquid(wd)
   Liquid(wc)
   Contained_Liquid(wd)
   Contained_Liquid(wc)
```

The activation of a process causes changes in the quantities involved in a situation: such changes are described through the sign of the derivative of the related quantities. This sign is "qualitatively calculated" by 'resolve_influences'. The qualitative nature of a description sometimes doesn't allow to execute that resolution without using heuristic information about interested domain.

```
heat(wd) is constant.
t_melt(wd) is constant.
heat(wc) is constant.
t_melt(wc) is constant.
amount_of(wd) is increasing.
temperature(wd) is constant.
amount_of(wc) is decreasing.
temperature(wc) is constant.
level(Contained_Liquid(wd)) is increasing.
level(Contained_Liquid(wc)) is decreasing.
volume(wd) is increasing.
volume(wc) is decreasing.
amount_of_in(d,water) is increasing.
amount_of_in(c,water) is decreasing.
pressure(wd) is increasing.
pressure(wc) is decreasing.
flow_rate(Liquid_Flow(wc,wd,pl)) is decreasing.
t_boil(wd) is increasing.
t_boil(wc) is decreasing.
```

The variation in parameter values causes changes in the current situation: these changes require the analysis of the new situation that results.
The purpose of 'limit_analysis', which ends the inferential cycle of a physical system situation's analysis, is to establish the new order of Quantity Spaces which will give rise to new Structures of CDs and the 'qualitative' duration of the situation.

```
Duration of situation S1 is INTERVAL

Limit Hypotheses found:

1-  amount_of_in(c,water)eq amount_of_in(d,water),
    pressure(wc)eq pressure(wd),
    a(flow_rate(Liquid_Flow(wc,wd,pl)))eq ZERO
2-  amount_of_in(c,water)eq amount_of_in(d,water),
    pressure(wc)eq pressure(wd)
3-  amount_of_in(c,water)eq amount_of_in(d,water),
    a(flow_rate(Liquid_Flow(wc,wd,pl)))eq ZERO
4-  amount_of_in(c,water)eq amount_of_in(d,water)
5-  pressure(wc)eq pressure(wd),
    a(flow_rate(Liquid_Flow(wc,wd,pl)))eq ZERO
6-  pressure(wc)eq pressure(wd)
7-  a(flow_rate(Liquid_Flow(wc,wd,pl)))eq ZERO

Hypothesis number: 1.  /* User's choice */

Quantity Spaces in situation S2:

amount_of_in(d,water)eq amount_of(wd)
amount_of_in(c,water)eq amount_of(wc)
a(temperature(wd))gt a(t_melt(wd))
a(t_boil(wd))gt a(temperature(wd))
```

```
a(temperature(wc))gt a(t_melt(wc))
a(t_boil(wc))gt a(temperature(wc))
amount_of_in(c,water)eq amount_of_in(d,water)
pressure(wc)eq pressure(wd)
a(amount_of(wd))gt ZERO
a(amount_of(wc))gt ZERO
a(amount_of_in(d,water))gt ZERO
a(amount_of_in(c,water))gt ZERO
flow_rate(Liquid_Flow(wc,wd,pl))eq ZERO
```

The following example examines the description of a system in which a Process of liquid outflow from a container can occur.
The scenario is as follows:

```
def(scenario) :: flow_out with
   (individuals: (c, d, p) and
    facts: ('Container'(c), 'Container'(d),
            'Fluid_path'(p)) and
   always: ('Fluid_connected'(inside(c), inside(d), p),
            'Open'(c),'Open'(d)) and
   individuals_in_situation: (c, d, p) and
   facts_in_situation:
       (a(amount_of_in(c, water)) eq 'ZERO',
        a(capacity(c)) gt 'ZERO',
        a(capacity(c)) gt a(capacity(d)),
        a(amount_of_in(d,water)) lt a(capacity(d)),
        a(amount_of_in(d,water)) gt a(capacity(c)))).
```

The flowing_out of liquid could again be modelled by 'Liquid_Flow', but in an extremely approximate way: therefore it's advisable to introduce a new process for the specific description of this physical phenomenon.

```
def(process) :: 'Flowout'(Can,Wc,P) with
   (individuals:
      ('Container'(Can),
       'Piece_of_Stuff'(Wc),
       active('Contained_Stuff'(Can,Subs,Wc))) and
    preconditions:
       active('Liquid_Flow'(Src,Wc,Path)) and
    q_conditions: a(amount_of(Wc)) eq capacity(Can) and
    relations: (introduces(P,'Piece_of_Stuff'),
                made_of(P) becomes made_of(Wc)) and
    influences:
      'I+'(amount_of(P),
           a(flow_rate('Liquid_Flow'(Src,Wc,Path))))).
```

The opportunity of completing incrementally the knowledge base about a physical domain emphasizes QPL's modularity.

4. DISCUSSION

At this point it may be interesting to understand what are the differences between our approach to QPT and the original one presented by K.D.Forbus in [6].
In particular we want to show how some fundamental concepts of the theory have found, in these two models, different (but related) expressions.

QPT is centered around the concept of change: a physical situation is described by reasoning about changes and their causes (processes).
The classic problem which arises in A.I. when dealing with dynamical descriptions is the **frame problem** [13], namely "when something happens, how to tell which facts remain true and which facts don't".
Forbus uses histories to solve the frame problem [Hayes].
Histories are descriptions of objects that extend through time but are always bounded spatially: they help in solving the frame problem because objects can interact only when their hystories intersect.
Forbus proposes QPT as a solution - in the qualitative dynamics - to the new problems arisen from this approach to the frame problem. These problems are:

- the ***local evolution*** problem (how histories are generated);
- the ***intersection / interaction*** problem (which intersections of histories actually correspond to interactions between the objects).

The solution lies in describing explicitly the changes and their causes: this leads to define the concepts of physical process and influence.

In Forbus' computational model the theoretical notion of history has been implemented using the concept of **slice**: again following Hayes, a slice of a history denotes a piece of an object's history at a particular time.
Objects are represented by individuals. There are two distinct but related notions of existence for an individual.

The first is *logical existence*, which simply means that a particular individual can exist given a certain state of affairs.

The second notion is *physical existence* which means that a particular individual actually exists at some particular time.
In Forbus' model, an individual which logically exists, can physically exist only if it has a slice associated to it.
For example the slice associated to an active process instance corresponds to the situation in which it has been activated.

Forbus calls ***situation*** the collection of slices corresponding to the objects that exist in it.
Thus a situation describes the collection of objects on which reasoning is done at a particular time.

In our work, we faced the *frame problem* when trying to build a new primitive for modeling actions on a physical system.
QPT does not provide tools to model actions on a system by external agents such as opening a valve, turning on a pump, introducing a ladle in a pan and so on.
The availability of such models could be very important when modeling industrial plants from the physical standpoint and when treating process control problems [2][5][18].
The three basic primitives mentioned earlier (prove, remember, forget) can be used to define an action in a way similar to the one found in STRIPS [16]:

```
def(action) :: <name> with
  (forget:   <list of sentences> and  /* delete list */
   remember: <list of sentences>).    /*    add list */
```

This model of action doesn't give rise to the frame problem *provided that*:

- after the execution of an action the basic deductions of QPT are used to derive the resulting situation. No explicit frame axioms are needed because the inference rules and domain descriptions embody or create the bindings which are necessary to establish all changes induced by the action in the situation. Thus QPT can be considered a theory of causal relationships (which P.Hayes retains necessary for any general solution to the problem itself [10]);

- the context in which the action takes place is circumscribed well enough for all its local effects to be expressable while leaving the task of analysing their propagation to the QPT deductions.

5. REMARKS

In the logic based QPT computational model presented here the concept of "logical truth" has taken the place of "physical existence".
The pair <logical existence, slice>, which represents physical existence, corresponds to a set of assertions derived from knowledge concerning the current physical existence.
Thus in our computational model a situation is the set of all true expressions present in a knowledge base.

Intuitively, the aim is to extend the concept of logical and physical existence to every QPL sentence.
For each sentence we can then define three basic logical operations:

- **remember** : a parametric sentence (not ground term) is instanced (it logically exists) and asserted among true sentences (it physically exists).
- **prove** : knowledge corresponding to true sentences is used to make deductions to infer new (further) knowledge.
- **forget** : a sentence stops to be true and to exist physically; therefore, it is removed from set of true sentences.

These operations, which are not explicit in Forbus' computational model, represent in our approach the basic level for deduction activities supported by QPL: QPT inferences are simply reviewed in these terms.

6.CONCLUSIONS

In this work an interpretation of Qualitative Process Theory in terms of logic programming has been proposed.
The major advantage provided by this approach can be found in the introduction of the notion of 'action' which allows an extension of the Sole Mechanism assumption, the kernel of Forbus' QPT computational model.

The computational model proposed here has been implemented in *Prolog-2*. It constitutes a running program which allowed the model to be tested by simulating a set of physical situations (industrial plants, liquid behaviors and so on) starting with their descriptions.
Currently, a refined release of the system which will allow the testing of its applicability in fields such as medicine and process control is under development.

ACKNOWLEDGMENTS

This work benefitted from valuable conversations with Prof.G.Degli Antoni. We are also indebted with M.Maiocchi and A.Beltramini for helpful advice and comments and with H.Taylor for his great contribution to the final release of the paper.

REFERENCES

[1] Bandini,S.,M.Bruschi,M.G.Filippini,A.Molesini
UN APPROCCIO PROLOG ALLA FISICA NAIVE: L'ESEMPIO DEI PROCESSI QUALITATIVI
to appear on Proceedings of Gulp,2 Rome 1988

[2] Beltramini,A., M.Maiocchi
MODELING INDUSTRIAL PLANTS AND THE RELATED CONTROL THROUGH PETRI NETS
University of Milan - R.I.Dept. of Computer Science - Milano, 1981

[3] Bratko,I.,I.Mozetic,N.Lavrac
AUTOMATIC SYNTHESIS AND COMPRESSION OF CARDIOLOGICAL KNOWLEDGE
in P.Hayes,D.Michie,Y.Richards (eds.)
"Machine Intelligence 11"
Oxford University Press,
Oxford 1986

[4] Bruschi,M.
MECHANO, UN AMBIENTE DI SVILUPPO DI PROGRAMMI PER IL RAGIONAMENTO QUALITATIVO SU SISTEMI FISICI: UNA APPLICAZIONE PER GLI IMPIANTI INDUSTRIALI
Tesi di Laurea in Scienze dell'Informazione, Università di Milano
A.A. 1986-87

[5] Filippini,M.G.
DESCRIZIONE E MODELLAZIONE DI IMPIANTI INDUSTRIALI E DEL RELATIVO CONTROLLO: UNA PROPOSTA DI LINGUAGGIO DI RAPPRESENTAZIONE
Tesi di Laurea in Scienze dell'Informazione, Università di Milano
A.A. 1986-87

[6] Forbus,K.D.
QUALITATIVE PROCESS THEORY (Technical Report 789)
MIT Artificial Intelligence Laboratory, 1984

[7] Hayes,P.J.
THE NAIVE PHYSICS MANIFESTO
in "Expert Systems in the Micro_electronic Age"
Edited by Donald Michie, 1979

[8] Hayes,P.J.
THE SECOND NAIVE PHYSICS MANIFESTO
in "Formal Theories of the Commonsense World"
Edited by Jerry R.Hobbs & Robert C.Moore - Ablex publishing Corporation
Norwood, New Jersey, 1985

[9] Hayes,P.J.
NAIVE PHYSICS I: ONTOLOGY FOR LIQUIDS
from [11]

[10] Hayes,P.J.
THE FRAME PROBLEM AND RELATED PROBLEMS IN ARTIFICIAL INTELLIGENCE
Readings in Artificial Intelligence
B.L.Webber, N.J.Nilsson (eds.),
TIOGA, Pu.Co., Palo Alto CA, 1981

[11] Hobbs,J.R.,R.C.Moore (eds.)
FORMAL THEORIES OF THE COMMONSENSE WORLD
Ablex publishing Corporation
Norwood, New Jersey

[12] Leaning,M.S.,E.Nicolosi,
MODEL - SOFTWARE FOR KNOWLEDGE_BASED MODELLING OF COMPARTMENTAL SYSTEMS,
Biomedical Measurements, Inf.Contr.,Vol.1, N°4,
1986

[13] McCarthy,J.,P.Hayes
SOME PHILOSOPHICAL PROBLEMS FROM THE STANDPOINT OF ARTIFICIAL INTELLIGENCE
Machine Intelligence 4, Edinburgh University Press, 1969

[14] McCord,M.,J.F.Sowa,W.G.Wilson,A.Walker (ed.)
KNOWLEDGE SYSTEMS AND PROLOG
Addison_Wesley Publishing Company, Inc.,1987

[15] Molesini,A.
UN MODELLO COMPUTAZIONALE PER UNA RAPPRESENTAZIONE DI IMPIANTI INDUSTRIALI E DEL RELATIVO CONTROLLO
Tesi di Laurea in Scienze dell'Informazione, Università di Milano
A.A. 1986-1987

[16] Nilsson,N.J.,R.Fikes
STRIPS: A NEW APPROACH TO THE APPLICATIONS OF THEOREM PROVING TO PROBLEM SOLVING
Artificial Intelligence 2, 1971

[17] Sterling,L.,E.Shapiro
THE ART OF PROLOG
Advanced Programming Techniques
The MIT Press - Cambridge, Massachusetts, 1986

[18] Stefanini,A.,G.Guida,M.Gallanti,L.Spampinato
REPRESENTING PROCEDURAL KNOWLEDGE IN EXPERT SYSTEMS: AN APPLICATION TO PROCESS CONTROL
in Proocedings of IJCAI, 1985

[19] Varsek,A.
QSIM IN PROLOG
Workshop on Qualitative Modelling
Ljubljana, 1987