5. Forbus,K.D., "Qualitative Reasoning about Space and Motion", in Mental Models, ed. Stevens,A. LEA Publishers, Hillsdale, N.J. (1983).
6. Giunchiglia,F. & Weyhrauch,R.W., "A Multi-Context Monotonic Axiomatization of Inessential Non-Monotonicity", in Maes,P. & Nardi,D., "Meta-level Architectures and Reflection", North-Holland, Amsterdam (1987).
7. Greeno,J.G., "Conceptual Entities", in Mental Models, ed. Stevens,A. LEA Publishers, Hillsdale, N.J. (1983).
8. Hayes,P.J., "The Naive Physics Manifesto", in Expert Systems in the Micro Eletronic Age, ed. Michie,D., Edinburgh University Press, Edinburgh, England (1979).
9. Minsky, M., "Why People Think Computers Can't", in The AI Magazine, Fall 1982
10. Norman,D.A., "Some Observations on Mental Models" in Mental Models, ed. Stevens,A. LEA Publishers, Hillsdale, N.J. (1983).
11. Palomba,L., "Modelli Qualitativi Per la Valutazione dell'Afferrabilita' di Oggetti", Tesi di Laurea, Dip. Scienze della Informazione, Milano (1987).
12. Piaget,J., "Psicologia ed Epistemologia", ed. Loescher, Torino (1974).
13. Weyhrauch,R., "Prolegomena to a Theory of Mechanized Formal Reasoning", in Artificial Intelligence vol. 13, n. 1, 2 (1980).

UN APPROCCIO PROLOG ALLA FISICA NAIVE: L'ESEMPIO DEI PROCESSI QUALITATIVI

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

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
Via Moretto da Brescia, 9 20133 Milano tel (02) 2772233

Sommario

In questo lavoro viene presentata una interpretazione Prolog del linguaggio proposto nella Teoria Qualitativa dei Processi (QPT) di K.D. Forbus, nell'ambito della Fisica Qualitativa.

Oltre alle caratteristiche principali del linguaggio della QPT ed ai corrispettivi costrutti Prolog, vengono messi in risalto alcuni aspetti originali di carattere integrativo a tale linguaggio (usando tecniche di metainterpretazione) che permettono una adeguata implementazione di un sistema Prolog per la realizzazione del paradigma computazionale della teoria qualitativa scelta.

1. INTRODUZIONE

Ottenere modelli computazionali del ragionamento di senso comune e qualitativo su sistemi fisici costituisce uno dei cardini di un nuovo settore dell'Intelligenza Artificiale, conosciuto con il termine Naive Physics (NP) [7] [8], che, specialmente in questi ultimi anni, sta suscitando sempre maggiore interesse negli operatori del settore.

Con Fisica Qualitativa, viene inoltre denotata quella specifica parte della NP dedicata alla definizione e all'impiego di modelli qualitativi legati al senso comune per descrivere situazioni in cui sono coinvolte l'interpretazione e l'interazione con una realtà fisica opportunamente rappresentata [1].

In questo paradigma, ha avuto un ruolo fondamentale il lavoro di K.D.Forbus [5]: l'approccio introdotto da questo autore è accentrato sulla nozione di *processo*, inteso intuitivamente come qualcosa che causa un cambiamento su oggetti nel tempo.

Viene proposta nei suoi lavori una teoria Qualitativa dei Processi (QPT) che definisce una nozione primitiva di processo fisico, utilizzabile per la modellazione di teorie della dinamica [6].

Scopo principale di questo lavoro è la presentazione di una interpretazione Prolog del linguaggio proposto nella QPT con l'intento di fornire la possibilità di descrivere situazioni fisiche.

Vengono cioè messi in risalto le peculiarità di Prolog, gli aspetti logici della QPT e una nuova interpretazione, favorita da Prolog stesso, dei costrutti di carattere prettamente inferenziale della teoria. L'attenzione dedicata all'utilizzo dei meccanismi inferenziali di Prolog e l'adozione di tecniche di metainterpretazione [9] [11] hanno permesso una realizzazione pressochè integrale del linguaggio della QPT.

Nel paragrafo seguente vengono presentate le caratteristiche principali del linguaggio in cui la QPT viene espressa. Seguono una presentazione delle motivazioni che hanno indotto alla scelta di Prolog per la realizzazione di un sistema che _ in termini di linguaggio logico _ presenti le stesse caratteristiche della QPT, una analisi delle scelte implementative e dei risultati ottenuti.

2. TEORIA QUALITATIVA DEI PROCESSI

2.1 Richiami

La Qualitative Process Theory (QPT) [6] si propone come strumento formale per la rappresentazione di conoscenze e di ragionamenti di senso comune sulla dinamica di situazioni fisiche.

Per *dinamica qualitativa* si intende una teoria qualitativa concernente i tipi di fenomeni che possono verificarsi in una situazione fisica.

La QPT utilizza un *linguaggio* formale di rappresentazione per esprimere proprietà e relazioni tra gli oggetti di un dominio di interesse, in particolare rispetto a una dinamica.

Per stabilire proprietà degli oggetti quali esistenza, attività o influenze, viene definito un insieme di *regole* che costituiscono la parte più prettamente inferenziale della QPT.

2.2 Gli oggetti base

Usando la QPT, diversi tipi di informazione devono essere specificati nello sviluppo di rappresentazioni, ovvero modelli, di domini. I costrutti sintattici funzionali a tale scopo verranno introdotti di seguito.

. *entità*

La modalità di rappresentazione di oggetti fisici è data dall'astrazione che chiameremo *individuo* o *entità*.

Ogni individuo in una situazione è sempre istanza di un particolare *tipo*; la dichiarativa

```
defEntity <type> <parts_list> <body>
```

indica che un'istanza di oggetto O di tipo `<type>` _ denotata dalla predicazione `<type>(O)` _ ha associata una lista di slot i quali ne costituiscono i parametri, e l'esistenza fisica di O implica la verità delle istanze di statement e regole contenute in `<body>`.

Segue un esempio di definizione di entità contenente alcune tipiche frasi nel linguaggio della QPT, per il cui significato si rimanda ad una trattazione completa della teoria [5].

```
/* linguaggio QPT: dichiarazione entità */
(defEntity piece_of_stuff ((amount_of quantity)
                           (volume quantity),
                           (pressure quantity),
                           (heat quantity),
                           (temperature quantity),
                           (t_boil quantity),
                           (location place))
   (Qprop+ (temperature ?self) (heat ?self))
   (Taxonomy (Liquid ?self) (Gas ?self)))
```

. *quantità*

L'astrazione usata per denotare tali parametri utilizza la nozione di *Quantity Space* [4] (discretizzazione della natura continua dei parametri). La dichiarativa

```
defQuantity_type <type>
```

indica che all'identificatore di funzione type applicato ad un oggetto devono essere associati due valori numerici, *Amount* e *Derivative*.

. *predicati e relazioni*

Identificatori di predicati e relazioni fanno capo a due distinte categorie sintattiche espresse mediante le dichiarative:

```
defPredicate <type> <body>
defRelation  <type> <body>
```

Statement e regole in `<body>` vengono utilizzati come descritto in defEntity qualora siano verificate istanze di predicati o relazioni.

. *costanti e fatti*

Una costante logica, come "water", viene introdotta dalla scrittura:

`(defineConstant water).`

La dichiarativa `defFact` intende dichiarare veri fatti generali relativi ad un certo dominio, come ad esempio `(defFact Substance water).`

. *descrizioni condizionali (DC)*

Per descrizioni condizionali si intendono *Individual View* (modellazione di individui il cui stato esistenziale può variare nel tempo) e *Processi* (descrizione time_dependent degli eventi che accadono nel tempo provocando cambiamenti).

Una DC è introdotta dalla dichiarativa `defView` o `defProcess` la cui sintassi è illustrata dal seguente esempio:

```
/* linguaggio QPT: dichiarazione processo */
(defProcess Liquid_flow (?src ?dst ?path)
   Individuals ((?src (Contained_Liquid ?src))
                (?dst (Contained_Liquid ?dst))
                (?path (Fluid_path ?path)
                       (Fluid_connected (location ?src)(location ?dst) ?path))
    Preconditions ((Aligned ?path))
    QuantityConditions ((Greater_Than (A(pressure ?src))
                                      (A(pressure ?dst)))
    Relations     ((Local flow_rate (Quantity flow_rate))
                   (Qprop+ flow_rate (- (pressure ?src)
                                        (pressure ?dst)))
    Influences    ((I+ (amount_of ?dst) (A flow_rate))
                   (I- (amount_of ?src) (A flow_rate))))
```

Il campo Influences riassume gli effetti diretti di un processo e non è quindi presente tra i campi di defView.

. *scenario*

E' costituito da individui coinvolti, relazioni tra di essi e condizioni iniziali espresse usualmente come disuguaglianze tra quantità.

La specificazione è denotata dalla scrittura:

`defScenario <name> <specifications>`, dove

`<specifications>` è una lista di campi che introducono individui, costanti, fatti ed una

descrizione della situazione iniziale.

2.3 Le inferenze base

Per essere utile, una rappresentazione deve supportare deduzioni.

Le deduzioni di base che coinvolgono i costrutti della QPT sono elencate di seguito.

. *elaborazione*

Consiste nella generazione di tutti i possibili individui ed istanze a partire da una collezione iniziale di oggetti. Tale generazione si rivela particolarmente utile nelle fasi di *envisioning* [3] eventualmente realizzabili sfruttando l'apparato teorico della QPT.

Dato un insieme di individui ed un vocabolario di Descrizioni Condizionali, la specificazione Individuals degli elementi del vocabolario è utilizzata per trovare collezioni di individui che possono partecipare ai vari tipi di DC, determinando *istanze* di view e processi che possono potenzialmente occorrere.

. *determinazione attività*

Rinvenire quali processi o istanze di view siano attive in una particolare situazione richiede di valutare la verità della congiunzione di condizioni nelle specificazioni Preconditions e QuantityConditions ad essi relative. Pertanto si dice che ogni istanza di Descrizione Condizionale, risultante dall'elaborazione, può trovarsi esattamente in uno degli stati *attivo* o *inattivo* in dipendenza da tale valore di verità.

La collezione delle istanze di processo attive in una situazione è detta *Process Structure* e rappresenta ciò che accade agli individui in quella situazione. Analoga definizione per *View Structure.*

Segue una descrizione non formalizzata dell'algoritmo per la realizzazione di tale inferenza:

```
Procedure DetActivity

   1. Sia Lista_DC l'insieme delle istanze ottenute dall'elaborazione.
   2. Per ogni istanza di DC in Lista_DC
        2.1 Si provino Preconditions e QuantityConditions,
        2.2 Se ne asserisca lo stato di attività.
```

. *determinazione cambiamenti*

Si esplica in una *risoluzione delle influenze* (descrizione degli effetti diretti dei processi) al fine di determinare cambiamenti in valore e segno delle derivate di parametri di oggetti.

Risolvere una quantità influenzata *direttamente* richiede la somma qualitativa delle influenze positive o negative su di essa, mentre per una quantità influenzata *indirettamente* occorre raggruppare gli statement di proporzionalità qualitativa che la riguardano.

. ***analisi dei limiti***

Rappresenta la determinazione delle possibili modifiche alle Strutture di Processi e View in una situazione, quando non siano più verificate le condizioni di attivazione di alcune istanze di Descrizioni Condizionali che di essa fanno parte.

3. UN APPROCCIO PROLOG ALLA QPT

La sezione seguente intende, in riferimento alla specifica applicazione, porre in rilievo le ragioni che hanno indotto a scegliere il linguaggio logico Prolog e presentare le scelte implementative proponendo un'analisi critica del linguaggio.

3.1 Perchè Prolog

La QPT si avvale di un linguaggio dichiarativo di tipo logico la cui sintassi può essere facilmente tradotta in *clausole* Prolog.

La costruzione di modelli computazionali per la QPT richiede l'ausilio di un meccanismo inferenziale: la scelta di Prolog permette di evitare l'implementazione di un meccanismo inferenziale per la specifica applicazione in quanto l'interprete Prolog è per sua natura tale.

L'istanziazione di uno degli oggetti base descritti _ ovvero l'istituzione di un legame dinamico tra il nome attuale dell'oggetto nella specifica situazione fisica e la descrizione parametrica della classe di appartenenza dello stesso _ è ottenuta in modo naturale attraverso *unificazione*.

3.2 Scelte implementative

Dal punto di vista della NP la QPT è un *cluster*, una collezione di conoscenze e di procedure (regole) di inferenza.

Un interprete per essa può essere visto come un modulo e, dal punto di vista implementativo, come *Abstract Data Type* [12]: le inferenze base rappresentano le procedure con cui eventuali *moduli_cliente* accedono alla struttura dati che costituisce la descrizione di un dominio.

Chiameremo questa struttura _ per sua natura referenziale _ Base Della Conoscenza *descrittiva* o *statica*.

Date queste premesse, è possibile ottenere l'architettura schematicamente rappresentata in Fig.1.

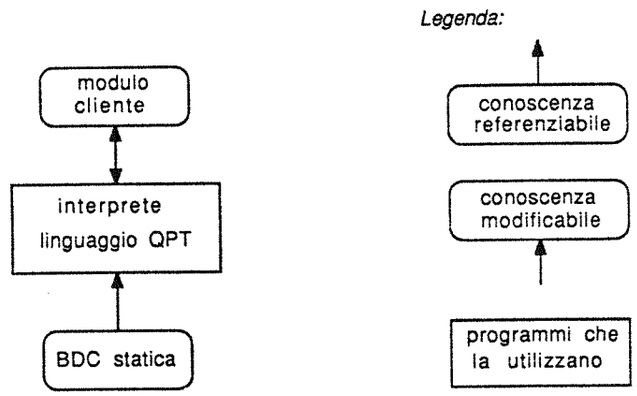

Fig. 1.

3.2.1 Base Della Conoscenza Statica

Contiene istanze delle dichiarative esposte nella sezione 2.2. Quanto segue costituisce il corrispettivo Prolog di alcuni esempi introdotti.

```
/* BDC Statica: fatti */
def(domain) :: fluid with
   (facts: ('Substance'(water))).

/* BDC Statica: dichiarazione entità */
def(entity) :: 'Piece_of_Stuff'(P) with
   (parts:       ('Quantity'(amount_of),
                  'Quantity'(volume),
                  'Quantity'(pressure),
                  'Quantity'(heat),
                  'Quantity'(temperature),
                  'Quantity'(t_boil),
                  'Place'(location)) and
      statements:(temperature(P) q_Dprop heat(P),
                  'Taxonomy'(('Liquid'(P),'Gas'(P))))).
```

```
/* BDC Statica: dichiarazione processo */
def(process) :: 'Liquid_Flow'(Src,Dst,Path) with
   (individuals:  ('Contained_Liquid'(Src),
                   '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)),
                   flow_rate q_Dprop (pressure(Src) - pressure(Dst))) and
    influences:   ('I+'(amount_of(Dst),a(flow_rate)),
                   'I-'(amount_of(Src),a(flow_rate)))).
```

Si osservi che i campi delle descrizioni sono espressi come combinazioni di frasi (cfr. **2.2 . *entità***) ciascuna delle quali fa capo ad una *categoria sintattica*. L'insieme di queste ultime costituisce una grammatica per tali frasi.

Struttura e recupero dell'informazione si ispirano al modello delle *property list* (strutture *frame_like*, cfr. **3.2.2 . *gestore di frame***).

3.2.2 Interprete per il linguaggio della QPT

Le scelte implementative, accanto all'adozione di Prolog come strumento per la traduzione del linguaggio della QPT, hanno comportato la delineazione di un'architettura per l'applicazione schematizzata in Fig.2.

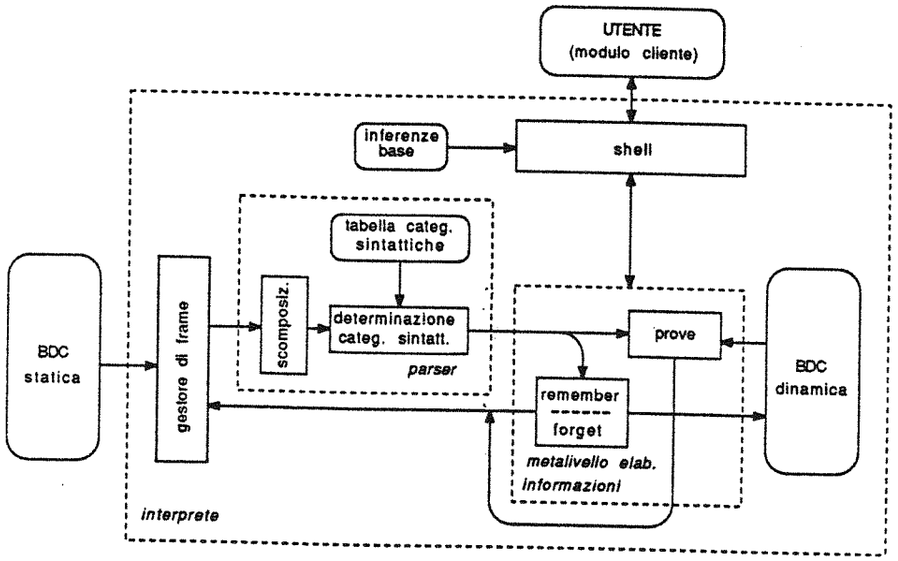

Fig. 2.

. *Base Della Conoscenza Dinamica*

La stesura di un programma Prolog deve partire dalla strutturazione logica della Base Della Conoscenza rispetto al problema in analisi.

Un modello computazionale per la QPT richiede di individuare un'ulteriore classe di conoscenza che diremo *istantanea* o *dinamica*.

La Base Della Conoscenza Dinamica risponde alla necessità di mantenere in una struttura dati l'insieme delle frasi vere in un certo istante, e come tale il suo contenuto _ rappresentazione della situazione corrente _ dipende dal tempo.

L'informazione non ha una struttura a frame (ma è espressa con normali termini Prolog) e pertanto il suo recupero è affidato a Prolog. La presenza dell'interprete per la QPT rende la Base Della Conoscenza Dinamica completamente invisibile all'utente anche dal punto di vista della modalità d'accesso. Inferenze e modifiche su di essa sono fornite come metalivello.

. *metalivello di elaborazione delle informazioni*

Fornisce le operazioni fondamentali che occorre eseguire sugli atomi (cfr. *parser*) costituenti le informazioni prelevate dalla Base Della Conoscenza Statica.

Tali operazioni sono distinte in:

remember, forget _ primitive di aggiornamento della Base Della Conoscenza Dinamica: mantengono i legami logici, specificati dalla conoscenza descrittiva, tra le informazioni presenti in essa.

Si osservi ad esempio la seguente dichiarazione di predicato:

```
def(predicate)::'Substance'(S) with
 (rules:
    (('Substance'(S),'Container'(C)) => 'Can_contain_Substance'(S,C)))).
```

La verità di 'Substance'(water) comporta l'asserzione, ad opera di *remember*, della regola QPT che compare nella definizione e la conseguente elaborazione delle eventuali informazioni ad essa associate.

```
/* BDC Dinamica: modifica */
remember S1 & S2 of Obj :- remember S1 of Obj, remember S2 of Obj.
remember Slot of Obj :- Slot of Any::Obj iS Value, kB_remember that Value.

forget S1 & S2 of Obj :- forget S1 of Obj, forget S2 of Obj.
forget Slot of Obj :- Slot of Any::Obj iS Value, kB_forget that Value.
```

prove _ costituisce il metalivello di deduzione. L'implementazione Prolog dell'interprete per la QPT propone l'uso di un differente metodo inferenziale su un pacchetto di costrutti linguistici fissato. Per effettuare le inferenze si è fornito un insieme di regole atte a costruire sul meccanismo inferenziale di Prolog il paradigma inferenziale della QPT, non essendo possibile esaurire _ mediante unificazione e risoluzione sulla base di conoscenza complessiva _ la verifica di ogni frase scritta nel linguaggio della QPT.

Ad esempio, provare la verità di un predicato richiede _ nel caso non sia asserito in Base Della Conoscenza Dinamica _ la verifica di un'eventuale regola QPT che lo implichi, oppure degli statement contenuti nel campo conditions _ se presente _ della definizione.

```
/* Paradigma inferenziale QPT: verifica di un predicato */
kB_prove a Obj of_type predicate :-
   Obj of_type predicate is_in_kB;
   kB_satisfy Obj of_type predicate;
   Obj of_type predicate has_true conditions.


/* BDC Dinamica: esecuzione */
prove S1 & S2 of Obj :- prove S1 of Obj, prove S2 of Obj.
prove Slot of Obj :- Slot of Any::Obj iS Part, kB_prove that Part.
```

. ***shell***

Riceve comandi provenienti da un generico utente (anche da eventuali moduli_cliente).

Essi possono essere di due tipi:

- caricamento descrizioni in Base Della Conoscenza Statica;
- effettuazione inferenze base (insieme di richieste al metalivello di elaborazione delle informazioni, codificate in una libreria di inferenze).

```
/* libreria inferenze: determinazione attività */
det_activity :-
   instances is_worth CD_list, /* CD_list è l'insieme delle istanze ottenute
                                  dall'elaborazione */
   for_each(member(CD_instance,CD_list),activate(CD_instance)).


activate(CD_instance) :-
   prove preconditions & q_conditions of CD_instance,
   kB_remember that CD_instance.
```

. ***gestore di frame***

Interfaccia la Base Della Conoscenza Statica ed il modulo metalivello di elaborazione delle informazioni che da essa richiede di prelevare informazioni.

Il recupero della conoscenza di interesse avviene specificando struttura frame_like e relativo campo da referenziare, come evidenziato dalla seguente query:

```
/* gestore di frame: query */
?- q_conditions of def(CatSynt)::'Liquid_Flow'(c,d,p1) iS Quantity_Conditions.

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

 /*  gestore di frame: recupero informazione da property list */
Attr of Obj iS Value :- Obj with Descr, Attr:Value is_in Descr.
Attr:Value is_in (Attr:Value and Others).
Attr:Value is_in (A:V and Others) :- Attr:Value is_in Others.
Attr:Value is_in Attr:Value.
Attr:null  is_in Any.
```

. ***parser***

L'informazione ottenuta dal gestore di frame viene scomposta dal parser nelle frasi costituenti o atomi. Di ogni frase viene riconosciuta la categoria sintattica, rispetto alla quale i programmi del metalivello di elaborazione delle informazioni sono parametrici.

```
/* parser: scomposizione ed analisi sintattica */
Elaborate that null.
Elaborate that (A';'B) :- Elaborate that A ; Elaborate that B.
Elaborate that (A','B) :- Elaborate that A , Elaborate that B.
Elaborate that Obj :- synt_type_of Obj iS Type, Elaborate a Obj of_type Type.
```

4. CONCLUSIONI E SVILUPPI FUTURI

Alla luce delle considerazioni di natura implementativa, si osserva come una facilitazione fornita da Prolog sia costituita dalla possibilità di orientare l'interprete verso una specifica applicazione mediante semplici regole costruite ad hoc: ciò ovvia l'inconveniente di una riformulazione totale o riscrittura ex novo di un meccanismo inferenziale dedicato.

Infatti nella nostra esperienza, Prolog, unitamente alla sovrastruttura elaborata, ha costituito un sistema flessibile e di proficuo utilizzo. Ad esempio, la struttura della Base della Conoscenza Statica è stata concepita in modo modulare grazie a strutture frame_like facilmente organizzabili e gestibili in questo sistema.

Inoltre la natura delle clausole Prolog si è rivelata adeguata alla traduzione di linguaggi formali vicini a quelli logici, di cui la QPT fornisce un esempio.

Dal punto di vista implementativo, gli impieghi possibili del sistema realizzato potrebbero essere modellazione di domini fisici specifici (ad esempio impianti industriali), simulazione, envisioning, interpretazione di misure, predizioni e diagnosi, planning, learning.

In questo lavoro è stato posto soprattutto l'accento sugli aspetti implementativi. Per ulteriori dettagli di carattere applicativo si veda [10].

E' peraltro in fase di sperimentazione _ a livello esemplificativo _ una applicazione rivolta alla simulazione di impianti industriali con tecniche di programmazione logica applicate a modelli qualitativi della specifica realtà in esame. Immediato sviluppo di quanto detto sarà un tentativo di check dei contenuti della Fisica Qualitativa.

A questo proposito, dal punto di vista della validazione di Modelli Computazionali per la Fisica Qualitativa espressi in Prolog, particolare attenzione verrà rivolta in senso comparativo ad altre esperienze analoghe: ci riferiamo in particolare ad esempi come quello di I.Bratko [2] (che riguarda la modellazione qualitativa del sistema elettrico del cuore) totalmente implementati in Prolog.

5. RINGRAZIAMENTI

Vogliamo qui ringraziare il Prof. G.Degli Antoni per i suggerimenti indispensabili per la definizione di questo approccio e per la supervisione del lavoro.

Un riconoscimento particolare va anche al Prof. M.Maiocchi e al Prof. A.Beltramini per gli spunti applicativi forniti.

Ad Antonio Disarò un grazie per il suo supporto nella fase implementativa.

6. BIBLIOGRAFIA

[1] Bandini,S., Naive Physics e suoi Modelli Computazionali, NIKE-TR-87-21-00, Istituto M. Negri, Milano, 1987

[2] Bratko,I.,Mozetic,N.Lavrac,Automatic Syntesis and Compression of Cardiological Knowledge, in P.Hayes, D.Michie, Y.Richards (eds.), "Machine Intelligence 11", Oxford University Press, Oxford, 1986

[3] De Kleer, J., J.S.Brown, The Origin, Form and Logic of Qualitative Physical Laws, Proc. 8th IJCAI, Karlsruhe, 1983

[4] De Kleer, J., J.S.Brown, Qualitative Reasoning Based on Confluences, Artificial Intelligence Journal, n.24, 1984

[5] Forbus, K.D., Qualitative Process Theory, Artificial Intelligence Journal, n.24, 1984

[6] Forbus, K.D., The Role of Qualitative Dynamics in Naive Physics, in J.R.Hobbs,R.C. Moore (eds.) Formal Theories of the Commonsense World, Ablex Pu. Co., Norwood, NJ, 1985

[7] Hayes, P.J., The Naive Physics Manifesto, in D.Michie (ed.) "Expert Systems in the Microelectronic Age", Edinburgh University Press, Edinburgh, 1979

[8] Hayes, P.J., The Second Naive Physics Manifesto, in J.R. Hobbs, R.C. Moore (eds.) Formal Theories of the Commonsense World, Ablex Pu.Co.,Norwood,NJ,1985

[9] McCord,M.,J.F.Sowa,W.G.Wilson, A.Walker (ed.), Knowledge Systems and Prolog, Addison_Wesley Publishing Company, Inc., 1987

[10] Molesini,A.,Un modello computazionale per una rappresentazione di impianti industriali e del relativo controllo, Dipartimento di Scienze dell'Informazione Milano, Tesi di Laurea A.A. 1986-87

[11] Sterling, L., E.Shapiro, The Art of Prolog: Advanced Programming Techniques, The M.I.T. Press, Cambridge, MA, 1986

[12] Wirth, N., Algorithms + Data Structures = Programs, Prentice-Hall,Inc. Englewood Cliffs,New Jersey,1976