LiTH-IDA-R-88-06

ISSN-0281-4250

INSTITUTIONEN FÖR DATAVETENSKAP

UNIVERSITETET OCH TEKNISKA HÖGSKOLAN LINKÖPING

A Representation of Coordinated Actions Characterized by Interval Valued Conditions

by

Christer Bäckström

RESEARCH REPORT

RKLLAB, March 1988

Postadress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden

Computer and Information Science Department
Linköping University

PhD theses:

* Available at: University of Microfilms Intl., 300 N. Zeeb Road, Ann Arbor, MI 48106, USA.

(*Linköping Studies in Science and Technology. Dissertations.*)

*No 77	**Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982.
*No 94	**Hans Lunell:** Code Generator Writing Systems, 1983.
*No 97	**Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983.
*No 109	**Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation, 1984.
*No 111	**Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984.
No 155	**Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons. 1987.
No 165	**James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning. 1987.
No 170	**Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems. 1987.

Licentiate of engineering Theses:

(*Linköping Studies in Science and Technology. Theses.*)

No 48	**Henrik Nordin:** On the Use of Typical Cases for Knowledge-Based Consultation and Teaching. 1985.
No 52	**Zebo Peng:** Steps Towards the Formalization of Designing VLSI Systems. 1985.
No 60	**Johan Fagerström:** Simulation and Evaluation of an Architecture based on Asynchronous Processes. 1985.
No 71	**Jalal Maleki:** ICONStraint, A Dependency Directed Constraint Maintenance System. 1987.
No 72	**Tony Larsson:** On the Specification and Verification of VLSI Systems. 1986.
No 73	**Ola Strömfors:** A Structure Editor for Documents and Programs. 1986.
No 74	**Christos Levcopoulos:** New Results about the Approximation Behaviour of the Greedy Triangulation. 1986.
No 104	**Shamsul I. Chowdhury:** Statistical Expert Systems - a Special Application Area for Knowledge-Based Computer Methodology. 1987.
No 108	**Rober Bilos:** Incremental Scanning and Token-based Editing. 1987
No 111	**Hans Block:** SPORT-SORT Sorting Algorithms and Sport Tournaments. 1987.
No 113	**Ralph Rönnquist:** Network and Lattice Based Approaches to the Representation of Knowledge. 1987.
No 118	**Mariam Kamkar, Nahid Shahmehri:** Affect-Chaining in Program Flow Analysis Applied to Queries of Programs. 1987
No 126	**Dan Strömberg:** Transfer and Distribution of Application Programs. 1987
No 127	**Kristian Sandahl:** Case Studies in Knowledge Acquisition, Migration and User Acceptance of Expert Systems. 1987

LiTH-IDA-R-88-06 ISSN-0281-4250

A Representation of Coordinated Actions Characterized by Interval Valued Conditions

by

Christer Bäckström

Internet: CBA@IDA.LIU.SE
UUCP: enea!liuida!cba

Abstract: We introduce the concept of action structures as a set of actions together with information about their relative temporal ordering. Actions are characterized by pre- and post-conditions stating how they change the world. They are also characterized by prevail- and keep-conditions, stating what they require and cause, respectively, to hold during their execution. This way, the definition of actions can implicitly specify when actions are allowed to or have to occur in parallel. It is also possible to let the definition of an action implicitly specify the terminating condition for the action. The conditions used to characterize actions are expressed as partial world states. The main novelty of this report is that these partial world states can range over discrete domains as well as over interval valued domains.

Postadress:
Institutionen för datavetenskap
Universitetet i Linköping och
Tekniska Högskolan
581 83 Linköping

Mailing address:
Department of Computer and
Information Science
Linköping University
S-581 83 Linköping, Sweden

1. Introduction

This report presents some contributions to the representation of *actions* and of *action structures* in particular. An action is something that happens in the world, and we won't distinguish between things that are brought about by an agent and things that happen without the direct participation of an agent, but simply use the word action for both cases. This means that the occurence of an earthquake will qualify perfectly well as an action, as well as the occurence of someone painting a wall will. Among the most important characteristics of actions are that they have a duration in time, that they can go on in parallel and that they usually affect the world. Action structures mean more or less complex systems of actions going on in sequence and/or in parallel and possibly interacting and/or interfering with each other. We will for the moment informally define an action structure as a set of actions together with some further information regarding e.g. their relative time order.

Action structures arise in many important application areas for AI, like e.g. autonomous vehicles, factory automation, natural language processing, office information systems, intelligent operating systems, reasoning about physical processes etc.

It is desirable that a practical system incorporating reasoning about action structures is based on a solid theoretical ground. This has not always been true for systems developed so far, and it is sometimes unclear what is actually going on inside those systems. We think that this calls for an increased effort in research about action structures, and especially how to formally represent them in order to get a clearer understanding of what is going on. Such a formalism must also be flexible enough to support different kinds of reasoning about action structures, such as planning, plan projection, replanning, plan execution, plan observation and plan recognition. Most of these topics appear in most of the application areas mentioned above, so we think that it is important to be able to use the same formalism for all of them.

Planning is probably the one of the above topics that has been given the most attention in AI research, and it has been a hot topic for at least 20 years. Despite this, almost all of the work done has used the simplifying assumptions that actions have no duration in time and that all actions go on in sequence, and they do therefore not suffice for our purpose. A good summary of the state of the art in this kind of planning has been written by Chapman [Cha87]. During the past 5 years or so, there has been an increased interest in parallel

actions, and some works in this area are discussed in section 11 of this report.

Some of the non-AI branches of computer science has been dealing with parallel actions for a long time in e.g. operating systems, shared data bases and concurrent programming languages. These applications do, however, almost alway use some kind of explicit synchronization or predefined order of execution for interfering actions, and they usually deal with a very limited set of possible actions.

The remainder of this report presents some contributions to the formal representation of parallel actions.

2. Basic ideas

Most of the action structure formalisms presented so far in AI have been expressed in logic on the syntactical level, and the semantics has usually been left unattended, except for first order logic solutions. Although logic has been the standard tool for most research in knowledge representation, it is not obvious that it is a good or even sufficient tool for representing action structures and many other AI problems. This problem has been discussed by Drew McDermott, who in a recent paper [McDer87] questions whether logic is powerful enough for temporal reasoning. He suggests that a lot of temporal reasoning problems like e.g the frame problem might be impossible to solve with logic, but rather require model theoretic solutions.

The formalism presented in this report is based on a representation of action structures that we presented in [Bäc88], and that work was based on a formalism developed by Erik Sandewall and Ralph Rönnquist [San86]. Like them, we will only define the semantics, and leave the question of language open, and we hope that this will lead to fewer problems than usually appear in logic formalisms for temporal reasoning. For further reading on this topic, see [San88]. Another feature with this approach is that there is a choice of what reasoning mechanism to use. An interesting idea would be to do reasoning by constraint prooagation on the semantical level instead of logical inference on the syntactical level.

The basic ideas in our previous approach are:

- An *action structure* is viewed as a set of actions, each of which has a start point and an end point, and where the set of all such points is partially

ordered for time.

- *Partial state* descriptions are used, where each *'feature'* or proposition in a partial state description can have a definite value (e.g. a truth value, or some other value from a discrete range) or be undefined. The relation of 'containing less information than' is defined between the partial states, and thus defines a lattice over the space of partial states.
- Each action is characterized by a *pre-condition*, a *post-condition*, a *keep-condition* and a *prevail-condition*, and each of these conditions is expressed as a partial state. The pre-condition and post-condition characterize what must hold at the beginning and end of the action, respectively. The prevail-condition characterizes what must hold for the whole duration of the action in order to execute the action successfully. The keep-condition, on the other hand, characterize what will hold for the whole duration of the action as a cause of its occurrence.

Although the pre- and post-conditions don't have to be different, they usually are and so expresses how the action changes the world. An action having a certain partial state as prevail-condition can only occur if it is temporally subsumed by an action having that same partial state as keep-condition. It is thus possible to express not only that two actions are **allowed** to occur in parallel but also that they are **required** to co-occur. This is important if two or more actions have to cooperate to succeed in achieveing a certain goal. Consider e.g. the scenario of changing gears in a car with a manual gearbox. Two actions are necessary to perform in this scenario, namely pressing down the clutch and changing the gear lever, respectively. These two actions can however not be performed in arbitrary order, but the gear lever must be changed while keeping the clutch pressed down (fig. 1).

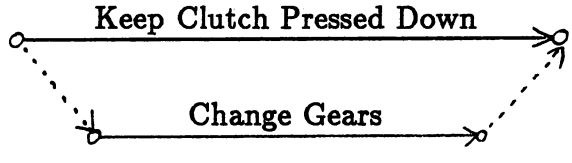

Fig. 1.

Any other temporal ordering between these two actions would fail the goal and, in many cases, even destroy the gearbox. The solution is to let the clutch-pressing action have as keep-condition that the clutch is pressed down,

and the gear-changing action would have as prevail-condition that the clutch is pressed down. This way we would enforce the temporal ordering of fig. 1.

3. Partial states

We assume that a finite number of aspects of the world under consideration will suffice to describe it. These aspects will be called features, and the combination of all the feature values at a certain point in time reflects the momentary state of the world. However, two temporally unrelated time points in two parallel trains of actions cannot both specify the world completely. For this reason, we will introduce the concept of partial states, which are understood as world states where some or all of the features are allowed to be undefined. Two action trains not affecting the same features of the world can thus occur in parallel without interfering, provided that the corresponding partial states of the time points in the trains do not define the same features.

The following section gives a formal definition of the discrete valued feature domains used in [Bäc88]. The two thereafter succeding sections motivates and formally defines the main topic of this report, namely interval valued feature domains. The accordingly revised formalism for partial states, actions and action structures is presented in sections 7 to 10.

4. Flat feature domains

It is assumed that the world under consideration can be modelled by a finite number of so called features. Every feature can take on any value from a corresponding predefined domain of values. The values of the feature domain can be any discrete range like e.g. the truth values domain {T, F}, the qualitative domain {+, 0, −} , the natural numbers or some subset thereof or a set of landmark values and the intermediate intervals. Each feature domain is also extended with the value u, denoting unknown, and, in order to form a complete lattice, the value k, denoting contradiction. A truth-valued feature domain is thus very much like a three-valued logic as described e.g. by Turner [Tur84]. A partial order $\sqsubseteq$ meaning 'contains less information than' is defined s.t. for any value x in the feature domain, the following relation holds:

$$u \sqsubseteq x \sqsubseteq k$$

We will assume no further requirements on the relation $\sqsubseteq$, and each feature domain will thus form a flat lattice, like the one in fig. 2. Feature domains of

this kind will be referred to as type A domains.

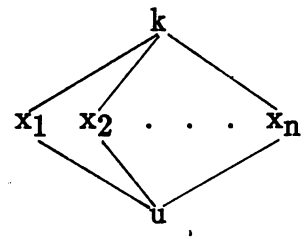

Fig. 2.

5. The need for interval valued features

Many problem areas cannot be modelled using only discrete valued feature domains of the type described above, but they rather require interval valued feature domains. Consider e.g. the scenario of road traffic, and assume a feature domain of speeds. A natural choice here could be to use a domain of landmarks for every 10 km/h and also including the open intervals in between. Such a domain can be considered as a discrete valued domain, although some of the elements are fixed values and others are intervals. Such a domain is, however, not sufficient. Suppose e.g. that there is an action expressing that a certain car is driving at a speed of say 50 km/h. It is in a practical case not very meaningful to have such an action since any implementation of it would most surely have an uncertainty in speed that can't be neglected. It would be better to define the action so that the speed of the car is between 45 and 55 km/h (or perhaps 40 and 50 km/h if the speed limit is 50 km/h and you are a very lawful driver). Problems will now arise if there is an action expressing that the car is accelerating from 0 to between 45 and 50 km/h. Such an action should have as pre-condition that the speed is 0 km/h and as post-condition that the speed is between 45 and 50 km/h. The keep-condition should be an interval expressing the speed during the acceleration, but this interval would reasonably be 0 to 50 km/h and thus overlapping with the 45 to 55 km/h interval.

There is thus a need for feature domains whose elements are, possibly overlapping, intervals. The solution to this problem is to allow non-flat lattices and to extend the ordering $\sqsubseteq$ to intervals. Two kinds of such lattices will be considered in this report, with the distinction being whether they contain only closed intervals or not.

6. Formalizing interval valued feature domains

We will first define the concept of intervals in the sense it is used in this report. We assume a set D which is totally ordered under $\langle \leq, < \rangle$, and we consider intervals over a subset S of D. The set S determines the modelling granularity and in the case that S = D we get the finest granularity possible. The closed interval [a:b] over $S \subseteq D$ is defined as

$$[a:b] = \{x \in D \mid a \leq x \leq b\} \text{ for } a,b \in S$$

the open interval (a:b) over $S \subseteq D$ is defined as

$$(a:b) = \{x \in D \mid a < x < b\} \text{ for } a,b \in S$$

and the semi-closed intervals (a:b] and [a:b) are defined in the obvious way. The set D will henceforth not be explicitly mentioned.

Let I and I' be intervals. The ordering relation $\sqsubseteq$ can now be defined for intervals s.t.

$$I \sqsubseteq I' \text{ iff } I \supseteq I'$$

which expands to the following inequalities

$$[a:b] \sqsubseteq [a':b'] \text{ iff } a \leq a' \text{ and } b' \leq b$$

$$(a:b) \sqsubseteq [a':b'] \text{ iff } a < a' \text{ and } b' < b$$

$$[a:b] \sqsubseteq (a:b] \sqsubseteq (a:b)$$

$$[a:b] \sqsubseteq [a:b) \sqsubseteq (a:b)$$

and we note that $\sqsubseteq$ still reflects information content.

We will now define the lattice of closed intervals over the set S defined above. Noting that a fixed value x is equivalent to the closed interval [x:x], we can form a domain S_1 of intervals over S in the following way.

$$S_1 = \{[a:b] \mid a \leq b \text{ and } a,b \in S\} \cup \{k\}$$

The element k is introduced in order to form a complete lattice. We will regard k as an abbreviation for the empty set and u as an abbreviation for the greatest interval in S_1 (i.e. u = [a:b]). Since the ordering relation $\sqsubseteq$ is defined as $\supseteq$, it will thus automatically extends to

$$u \sqsubseteq x \sqsubseteq k \text{ for any } x \text{ in } S_1$$

and $\langle S_1, \sqsubseteq \rangle$ forms a complete lattice with a top element k and a bottom element u.

Lattices of this kind will be referred to as type B feature domains.

Fig. 3 shows the type B lattice over the set {a, b, c} where a < b < c.
The lattice over S containing not only closed but also open and semi-closed intervals is defined as

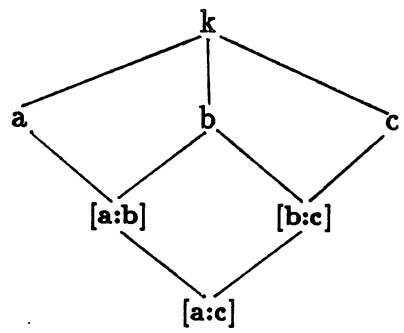

Fig. 3.

$$
\begin{aligned}
S_2 = \ & \{[a{:}b] \mid a \leq b \text{ and } a,b \in S\} \cup \\
& \{(a{:}b] \mid a \leq b \text{ and } a,b \in S\} \cup \\
& \{[a{:}b) \mid a \leq b \text{ and } a,b \in S\} \cup \\
& \{(a{:}b) \mid a \leq b \text{ and } a,b \in S\} \cup \{k\}
\end{aligned}
$$

The ordering relation is as usually defined as $\supseteq$ and thus it will automatically extend as in the previous case. Lattices of this kind forms type C feature domains.

The type C lattice over the same domain as the previous example is shown in fig. 4.

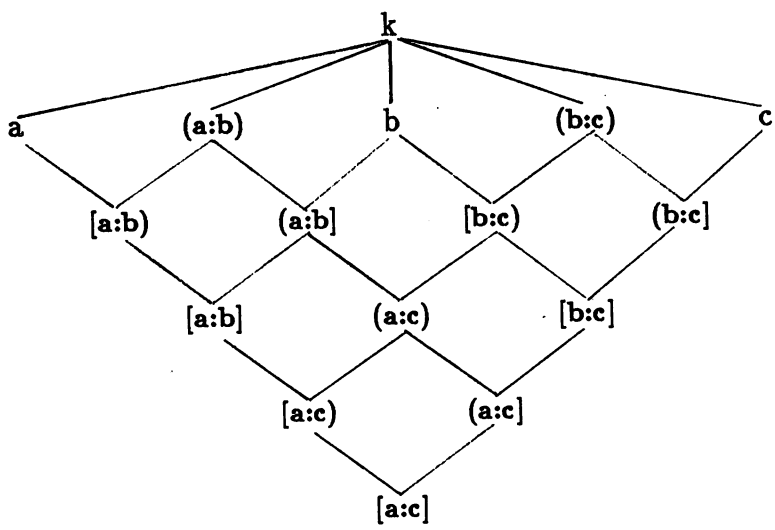

Fig. 4.

We do of course not require the domains, over which we define our interval

lattices, to be linear. A type C lattice could e.g. be defined over the domain $\{-\infty, -7, -1, 0, 1, 7, \infty\}$. In this case, we might, however, not want to represent the values $-\infty$ and ∞ but only the open intervals from $-\infty$ to -7 and from 7 to ∞. This would work fine and the bottom element of the lattice would thus be an open interval rather that a closed one.

7. Formalization of partial states

Letting F_i denote the i:th feature domain, the set of partial states of the world can now be expressed as the cartesian product

$$S = F_1 \times F_2 \times \ldots \times F_n$$

and the elements of S will be written as

$$\langle x_1, x_2, \ldots, x_n \rangle$$

The feature domain types may appear mixed in the partial states, so each of the F_i is allowed to be any of the types A, B or C.

Let s be a partial state in the domain S just defined. The element from domain F_i used to form s is denoted s[i] and is called the *projection* of s into the dimension i. The relation $\sqsubseteq$ is now extended to the space of partial states in the following way.

$$s \sqsubseteq s'$$

iff for every i

$$s[i] \sqsubseteq s'[i]$$

and it will thus still be an ordering reflecting information content. The structure $\langle S, \sqsubseteq \rangle$ now constitutes the lattice of partial states. The $\sqcup$ and $\sqcap$ will denote the usual lattice operations join and meet and $\top$ and $\bot$ will denote the top and bottom elements of the lattice, respectively.

The partial state s is said to *have the i:th feature* iff $s[i] \neq u$. The set of all i s.t. s has the i:th feature is denoted dim(s).

Two partial states s and s' are *co-dimensional* iff

$$\dim(s) = \dim(s')$$

and they are *anti-dimensional* iff

$$\dim(s) \cap \dim(s') = \emptyset$$

The partial state s is said to be *consistent* iff

$$s[i] \neq k \text{ for all } i.$$

8. Actions

We let V denote the set of operations. The inner structure of operations will not be considered here, but an operation could be seen as some kind of reference to an action class. Operations will however correspond to things that can be performed by an agent or that can occur, like e.g. "Press down clutch", "Earthquake" or "Play a C-chord".

An action can now be defined as a five-tuple

$\langle v,f,b,g,e \rangle$

where v is an operation, f is the prevail-condition, b is the pre-condition, g is the keep-condition and e is the post-condition.

The intuitive interpretation of these conditions is as follows. An action can begin whenever its pre-condition holds, and it will furthermore execute succesfully if its prevail-condition holds at all inner time points of the action. If an action so executes successfully, its post-condition will hold at the end point and the keep-condition will hold at all inner time points. In simple terms this means that the pre- and post-conditions express how the action changes the world. The prevail-condition expresses what the actions requires from the world during its execution, and the keep-condition expresses what it 'supplies' to the world during its execution.

We will focus our interest on actions where

- b and e are co-dimensional and
- $\dim(g) \subseteq \dim(b)$ and
- b and f are anti-dimensional

and we will call such actions valid. We let A denote the set of valid actions.

For interval valued features serving as qualitative descriptions of some continous parameter, we might also want a continuity restriction on the conditions. Consider e.g. the action of accelerating a car from 0 to 50 km/h. The pre- and post-conditions would simply have the values 0 and 50, respectively. The keep-condition should illustrate that the car is moving during the whole acceleration, but that it has not reached 50 km/h, and the closed interval [0:50] is not precise enough to express this. On the other hand, we can't let the keep-condition be e.g. the interval [5:45], since the speed of the car would then have to change instantaneously from 0 to 5 km/h. Consequently, the only reasonable value for the keep-condition is the open interval (0:50). Although the acceleration actions is best characterized in this way, there are

other kinds of action that require different keep-conditions. The continuity requirement is therefore not quite simple, and it will not be further discussed in this report.

There are three categories of actions that deserve a particular interest, namely:

I Actions that change the world, but where we know nothing about the changed feature during the execution of the action. 'Paint the wall green' is an example of such an action. These actions have

$b \neq e$ and $g = \perp$

II Actions that hold a feature constant at a value that would otherwise be unstable, like e.g. 'Keep the clutch pressed down' or 'keep the temperature at 295 K'. These actions have

$g \neq \perp$, $g \neq b$ and $g \neq e$

III Actions that change the world and where the change can be sharply defined and so define the actual termination of the action. Such a sharp change can often be defined by a logical sensor, e.g. by thresholding a value, if it isn't naturally present. An example of such an action is 'Heat the water until it starts boiling'. These actions have

$b = g$, $e \neq g$, $g \neq \perp$ and $e \neq \perp$

There is also a fourth very important class of actions of the form

$\langle \text{Persist}, \perp, s, s, s \rangle$

and these actions are called persistence actions. The persistence actions are used to carry persistence information between nodes in the action structure, and they are defined for all s from some predefined subset of the set S of partial states. We cannot usually allow persistence actions for all partial states s, since some feature values are unstable and cannot persist, e.g. the down state of a clutch. We must also require that s is consistent and that $s \neq \perp$. The Persist action corresponds to the Noop actions used in [San86], with the difference that the persistence information is carried in the keep condition instead of in the prevail-condition. Using the keep-condition makes the formalism clearer, but is also intuitive in the sense that a Persistence action could actually be implemented as an action that monitors that s really persists. The use of Persistence actions will be explained below.

9. Action structures

In order to formalize the concept of action structures, we assume a set T of time points. A partial ordering $\leq$ over T denotes temporal precedence between time points. An inequality relation $\neq$ is further used to distinguish between necessarily different elements of T.

An action structure over a set A of valid actions is defined as a quadruple

$\langle T, \leq, \neq, P \rangle$

where P, the plan, is a set of triples of the form

$\langle t, a, t' \rangle$

s.t. a is an action in A, and t and t' are time points in T denoting the start and end points of the action. We must, of course, require that

$t \leq t'$ and $t \neq t'$

In expanded form the triple looks like

$\langle t, \langle v,f,b,g,e \rangle, t' \rangle$

10. Admissable action structures

We will now define the concept of *admissable* action structures, which have the useful property that for any total ordering of T that is a strengthening of the orderings $\leq$ and $\neq$, the action structure is guaranteed to execute correctly. We will start with some other definitions leading forward to the definition of admissability.

The *incoming actions* for a time point t are those actions ending at that time point, or more formally all actions a s.t. there is an action occurence $\langle t',a,t \rangle$. The *outgoing actions* for a time point t are similarily all actions a s.t. there is an action occurence $\langle t,a,t' \rangle$.

We also let $s^-(t)$ denote the join of the post-conditions of the incoming actions for the time point t. $s^+(t)$ similarily denotes the join of the pre-conditions of the outgoing actions for the time point t.

An action structure is *coherent* if, for every time point t in T

1. the post-conditions of the incoming actions are consistent and anti-dimensional,
2. the pre-conditions of the outgoing actions are consistent and anti-dimensional,

3. if the time point has both incoming and outgoing actions, then
   a) $s^-(t)$ and $s^+(t)$ are co-dimensional and
   b) $s^+(t) \sqsubseteq s^-(t)$

An action structure $\langle T,\leq,\neq,P\rangle$ is also coherent if there exists a coherent action structure $\langle T,\leq,\neq,P'\rangle$ s.t. $P \subseteq P'$ and all members of $P' - P$ are persistence actions.

Note that if feature domain i is of type A, then point 3 above collapses to

$$s^-(t)[i] = s^+(t)[i]$$

for that domain, which corresponds to the definition in [Bäc88]. For type B and C domains, on the other hand, point 3 says that an outgoing action cannot require a more specific interval than is supplied by some incoming action. Suppose e.g. that there is an action terminating at t and which has as post-condition that a certain parameter is between 3 and 7. It would, in this situation, hardly be safe to start an action having as pre-condition that that same feature is between 5 and 8. If, on the other hand, the action has as pre-condition that the feature is between 2 and 7, it would be perfectly fine to allow it to start at t.

Coherence is not a sufficient condition for an action structure to have an execution since it only deals with the pre- and post-conditions, i.e. how actions change the world before and after their execution. Interference and cooperation between possibly parallel actions must also be taken into account, and this will be expressed by the alignment requirement below. This requirement intuitively says that an action having a certain feature value as a prevail-condition must be temporally subsumed by another action or chain of actions having a keep-condition that is at least as specific for that same feature.

A chain
$\langle t_0,\langle v_1,f_1,b_1,g_1,e_1\rangle,t_1\rangle,\ldots,\langle t_{n-1},\langle v_n,b_n,g_n,e_n\rangle,t_n\rangle$ of action occurences is said to *subsume* another action occurence $\langle t',\langle v',f',b',g',e'\rangle,t''\rangle$ in the i:th feature iff

$t_0 \leq t'$ and
$t'' \leq t_n$ and
$f'[i] \sqsubseteq g_k[i]$ for $1 \leq k \leq n$ and
$f'[i] \sqsubseteq b_k[i]$ for $2 \leq k \leq n$

An action structure $\langle T,\leq,\neq, P\rangle$ is *aligned* for the i:th feature iff there is a set $P' \subseteq P$ s.t. P' is a chain and every action occurence whose f, b, e or g has the i:th feature is either a member of P' or subsumed in the i:th feature by some chain $P'' \subseteq P'$.

An action structure $\langle T,\leq,\neq, P\rangle$ is admissable iff there is a $P' \supseteq P$ s.t. $P' - P$ contains only persistence actions and $\langle T,\leq,\neq,P'\rangle$ is both coherent and aligned for every feature.

The main topic of this report is to introduce interval valued feature domains to the formalism in [Bäc88] and to redefine the definitions of action structures and admissability accordingly. Discussion and examples of the use of action structures will therefore not be given here, and the interested reader is urged to consult [Bäc88] for further information on this topic.

11. Related Works

Temporal logics are usually very powerful but also too complex for temporal reasoning in many AI applications. The most referenced early works on temporal logics for AI have been done by Drew McDermott [McDer82] and James Allen [All84], but neither of them gives formal descriptions of the underlying semantics. A recent paper by Yoav Shoham [Sho87] presents a termporal logic with a formal semantics, and which subsumes both Allens and McDermotts systems, while also being a cleaner and more powerful approach.

Both Mike Georgeff [Geo86] and Amy Lansky [Lan86] have presented formalisms for events and causality. In both formalisms, events can be simultaneous but have no duration in time so it is not possible to express that an event occurs during another event.

Edwin Pednault [Ped86] expresses structures of parallel actions in an ordinary sequential planning formalism. Functions are used to describe how parameters change over time, and the history can be described by a sequence of time points at which any such function is replaced. Our approach provides more advanced synchronization and supports actions hierarchies. Pednaults formalism, on the other hand, can represent continous change, which is probably indispensable to many AI applications.

An early work was carried out by Gary Hendrix [Hen73], but his system was mainly a plan execution simulator, and there was no formal description except for some pieces of program code. His actions were STRIPS-like operators, but augmented with two extra conditions. One of these expresses how some parameter will change over time during the action and the other expresses the conditions for termination.

Acknowledgements

Many thanks to Erik Sandewall and Ralph Rönnquist for invaluable comments, ideas and discussions.

References:

[All84] **Allen James:** "Towards a General Theory of Action and Time", AI Journal 23, pp. 123 - 154, North-Holland 1984.

[Bäc88] **Bäckström Christer:** "A Representation of Coordinated Actions", Proc. SCAI'88 (Scandinavian Conference on AI), Tromso 1988.

[Cha87] **Chapman David:** "Planning for Conjunctive Goals", AI Journal 32, pp. 333 - 377, North-Holland 1987.

[Geo86] **Georgeff Michael:** "The Representation of Events in Multiagent Domains", Proc. AAAI-86, Philadelphia 1986.

[Hen73] **Hendrix Gary:** "Modeling Simultaneous Actions and Continous Processes", AI Journal 4, pp. 145 - 180, North-Holland 1973

[Lan86] **Lansky Amy:** "A Representation of Parallel Activity Based on Events, Structures, and Causality", SRI Technical Note 401, SRI 1986.

[McDer82] **McDermott Drew:** "A Temporal Logic for Reasoning About Processes and Plans", Cognitive Science 6, pp. 101 - 155, 1982.

[McDer87] **McDermott Drew:** "AI, Logic and the Frame Problem",, Proc. Workshop on the Frame Problem, Lawrence 1987, pp. 105 - 118.

[Ped86] **Pednault Edwin:** "Formulating Multiagent Dynamic-World Problems in the Categorical Planning Framework", Proc. Workshop on Reasoning About Actions Plans, pp. 47 - 82, Timberline 1986.

[San86] **Sandewall Erik, Rönnquist Ralph:** "A Representation of Action Structures",, Research Report LITH-IDA-R-86-13, Linköping University 1986. Also in Proc. AAAI-86, Philadelphia 1986.

[San88] **Sandewall Erik:** "Formal Semantics for Reasoning about Change with Ramified Causal Minimization", Internal Department Report, Dept. of Computer Science, Linköping University 1988.

[Sho87] **Shoham Yoav:** "Temporal Logics in AI: Semantical and Ontological Considerations", AI Journal 33, pp. 89 - 104, North-Holland 1987.

[Tur84] **Turner Raymond:** "Logics for Artificial Intelligence", Chapter 3, Ellis Horwood, West Sussex 1984.

Avdelning, institution, fakultet
Division, department, faculty

Department of Computer and
Information Science
Linköping University

ISBN:

ISSN: ISSN-0281-4250

Rapportnr:
Report no: LiTH-IDA-R-88-06

Upplagans storlek:
Number of copies:

Datum:
Date: March 1988

Projekt:
Project:

Titel:
Title: A Representation of Coordinated Actions Characterized by Interval Valued Conditions

Författare:
Author Christer Bäckström

Uppdragsgivare:
Commissioned by:

Dnr.:
Call no:

Rapporttyp:
Kind of report:

- ☐ Examensarbete/Final project
- ☐ Delrapport/Progress report
- ☐ Reserapport/Travel report
- ☐ Slutrapport/Final report
- ☒ Övrig rapport/Other kind of report

Rapportspråk:
Language:

- ☐ Svenska/Swedish
- ☒ Engelska/English
- ☐ _____

Sammanfattning (högst 150 ord):
Abstract (150 words):

We introduce the concept of action structures as a set of actions together with information about their relative temporal ordering. Actions are characterized by pre- and post-conditions stating how they change the world. They are also characterized by prevail- and keep-conditions, stating what they require and cause, respectively, to hold during their execution. This way, the definition of actions can implicitly specify when actions are allowed to or have to occur in parallel. It is also possible to let the definition of an action implicitly specify the terminating condition for the action. The conditions used to characterize actions are expressed as partial world states. The main novelty of this report is that these partial world states can range over discrete domains as well as over interval valued domains.

Nyckelord (högst 8):
Keywords (8):

Bibliotekets anteckningar

A Selection of Previous Research Reports.

LiTH-IDA-R-87-24 **Sven Moen:** Drawing Dynamic Trees.

LiTH-IDA-R-87-22 **Lars Ahrenberg, Arne Jönsson:** An Interactive System for Tagging Dialogues. To appear in *Proc. of 14th Conference of the Association for Literary and Linguistic Computing*, Göteborg 1-5 June, 1987.

LiTH-IDA-R-87-21 **Harold W. Lawson, Jr.:** Challenges and Directions in Computers and Education.

LiTH-IDA-R-87-20 **Krzysztof Kuchcinski, Zebo Peng:** Parallelism Extraction from Sequential Programs for VLSI Applications. This paper is to appear in *Microprocessing and Microprogramming, the Euromicro Journal*, 1988.

LiTH-IDA-R-87-19 **Zebo Peng:** Construction of Asynchronous Concurrent Systems from their Behavioral Specifications. Also in *Proc. of the 10th World Computer Congress*, Dublin, Ireland, Sept. 1-5, 1986.

LiTH-IDA-R-87-18 **Henrik Nordin:** Reuse and Maintenance Techniques in Knowledge-Based Systems.

LiTH-IDA-R-87-17 **Tony Larsson:** Specification and Verification of VLSI Systems Actional Behaviour This is a close version of a paper presented at *the 8th international conference on Computer Hardware Description Languages, CHDL*, 87.

LiTH-IDA-R-87-16 **Shamsul I. Chowdhury:** State of the Art in Statistical Expert Systems. Also in *Proc. of the Conference on Expert Systems and their Applications*, Avignon, May 1987.

LiTH-IDA-R-87-15 **Nils Dahlbäck:** Kommunikation med datorer i naturligt språk - vad är det och vem behöver det?

LiTH-IDA-R-87-14 **Peter Fritzson:** Window System Architectures - an Overview. Also presented at *A Tutorial on Software Development Environments*, Linköping University, Nov 24-25 1986.

LiTH-IDA-R-87-13 **Mats Wirén:** A Comparison of Rule-Invocation Strategies in Context-Free Chart Parsing. Also in *Proc. of the Third Conference of the European Chapter of the Association for Computational Linguistics*, Copenhagen, April 1—3, 1987.

LiTH-IDA-R-87-11 **Sture Hägglund, Christer Hansson, Tomas Sokolnicki:** Knowledge-Based Training of Case Management Routines and Emergency Procedures. Also in *Proc. of the 3rd Int. Conf. on Expert Systems*, London, 1987.

LiTH-IDA-R-87-10 **Andrzej Lingas:** On Parallel Complexity of the Subgraph Isomorphism Problem.

LiTH-IDA-R-87-09 **Andrzej Lingas, Marek Karpinski:** Subtree Isomorphism and Bipartite Perfect Matching are Mutually NC Reducible.

LiTH-IDA-R-87-08 **Christos Levcopoulos, Andrzej Lingas, Jörg-R. Sack:** Nearly Optimal Heuristics for Binary Search Trees with Geometric Generalizations. Also in *Proc of ICALP'87*, Karlsruhe, July 1987.

LiTH-IDA-R-87-07 **Peter Haneclou:** A Formal Approach to Reason-maintenance Based on Abstract Domains.

LiTH-IDA-R-87-06 **Johan Hultman:** COPPS - A Software System for Defining and Controlling Actions in a Mechanical System.

LiTH-IDA-R-87-05 **Christer Bäckström:** Logical Modelling of Simplified Geometrical Objects and Mechanical Assembly Processes.

LiTH-IDA-R-87-03 **Mikael R.K. Patel:** A Threaded Interpretive Language Supporting Programming in the Large. Also in *Proc. of The sixth Rochester Forth Conference*, University of Rochester, Rochester, New York, June 11-14, 1986.

LiTH-IDA-R-87-02 **Rober Bilos:** A Token-Based Syntax Sensitive Editor. Also presented at the *Workshop on Programming Environments - Programming Paradigms*, Roskilde, Denmark, October 22-24, 1986.

LiTH-IDA-R-87-01 **Wlodzimierz Drabent:** Do Logic Programs Resemble Programs in Conventional Languages? Also in *Proc of the Fourth IEEE Symposium on Logic Programming*, San Francisco, Aug. 31- Sept. 4, 1987.

The Department of Computer and Information Science
Linköping University

organizes undergraduate and graduate studies in *Computer Science*, *Telecommunication and Computer Systems*, and *Administrative Data Processing*. Research activities have an emphasis on advanced software technology and computer systems design and are organized in a number of research laboratories:

- *ACTLAB - Laboratory for Complexity of Algorithms*, which is concerned with the design and analysis of efficient sequential and parallel algorithms, and complexity theory, especially in the areas of computational geometry, data structures on bounded domains and graph algorithms.
- *AIELAB - Artificial Intelligence Environments Laboratory,* which conducts research on representation and manipulation of knowledge, problem solving techniques and expert systems with mechanical engineering applications.
- *ASLAB - Application Systems Laboratory,* which studies design of advanced support systems for interactive use of computers, including tools for automated construction of applications software.
- *CADLAB - Laboratory for Computer-Aided Design of Electronics*, which concentrates its research activities around tools for integrated development of hardware and software, graphics-based modelling and simulation techniques.
- *LIBLAB - Laboratory for Library and Information Science,* which studies methods for access to documents and the information contained in the documents, concentrating on catalogs and bibliographic representations, and on the human factors of library use.
- *LOGPRO - Laboratory for Logic Programming*, which concentrates its research activities around foundations of logic programming, relations to other programming paradigms and methodology.
- *NLPLAB - Natural Language Processing Laboratory*, which conducts research related to the development and use of natural language interfaces to computer software.
- *PELAB - Programming Environments Laboratory*, which works with design of tools for software development, specific functions in such tools and theoretical aspects of programs under construction.
- *RKLLAB - Laboratory for Representation of Knowledge in Logic*, which covers issues and techniques such as non-monotonic logic, probabilistic logic, modal logic and truth maintenance algorithms and systems.

Research Reports 1987 and 1988

LiTH-IDA-R-88-06	**Christer Bäckström:** A Representation of Coordinated Actions Characterized by Interval Valued Conditions.
LiTH-IDA-R-88-05	**Christer Bäckström:** Keeping and Forcing: How to Represent Cooperating Actions.
LiTH-IDA-R-88-04	**Wlodek Drabent, Simin Nadjm-Tehrani, Jan Maluszynski:** Algorithmic Debugging with Assertions.
LiTH-IDA-R-88-03	**Staffan Bonnier, Jan Maluszynski:** Towards a Clean Amalgamation of Logic Programs with External Procedures.
LiTH-IDA-R-88-02	**Ulf Nilsson:** Inferring Restricted AND-Parallelism in Logic Programs using Abstract Interpretation.
LiTH-IDA-R-88-01	**Jacek Leszczylowski, Staffan Bonnier, Jan Maluszynski:** Logic P[illegible]ucing S-Unification. A sl[illegible]*essing Letters*, North H[illegible]
LiTH-IDA-R-87-26	J[illegible]g System to Control P[illegible]