A framework for the conceptual modeling of discrete dynamic systems

A Framework for the Conceptual Modeling of Discrete Dynamic Systems.

by Jan L.G. Dietz
Kees M. van Hee

86/05

October 1986

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
All rights reserved
editor: F.A.J. van Neerven

A FRAMEWORK FOR THE CONCEPTUAL MODELING OF DISCRETE DYNAMIC SYSTEMS

Jan L.G. Dietz
Dept. of Industrial Engineering

Kees M. van Hee
Dept. of Computing Science

Eindhoven University of Technology
P.O. Box 513
5600 MB EINDHOVEN, The Netherlands

Universes of Discourse are representatives of the class of discrete dynamic systems. The complete and precise description of a Universe of Discourse therefore agrees with the specification of the structural and behavioural knowledge of a discrete dynamic system.
In this paper a framework is developed for the specification of discrete dynamic systems, including aggregates of communicating systems.
Ways of knowledge acquisition are discussed, leading also to the distinction of several typical kinds of systems. Among them are information systems.
Finally a first order language is reviewed as a vehicle for expressing knowledge about a discrete dynamic system.
The paper concludes with the presentation of some illustrative examples.

1. Introduction

The, commonly accepted, observation being made that an information system acquires, contains and provides knowledge about some other system, one is faced with the problem of adequately modeling this latter system such that the knowledge about it can be clearly defined and precisely expressed.
In the past years a lot of research has been devoted to systems modeling, particularly to the conceptual modeling and formal specifying of system dynamics (e.g. [StHo85], [SoKu85], [Ze82], [VeFu85], [MyWo80]). The research on which this paper reports also falls in this category. It presents a general framework in which the distinct kinds and degrees of knowledge can be discussed clearly and easily. We have called the framework SMARTIE and the kind of systems it is meant to deal with: discrete dynamic systems.
A discrete dynamic system is a system that is at each moment in one of a set of states. At some moments it performs a transition to a, not necessarily different, new state. A transition is triggered by a set of stimuli, called actions, that are exercised on the system at a particular moment. A transition is supposed to be executed instantaneously. The new state only depends on the present state and the set of actions. At the same time the system may produce a set of reactions, which may be considered as stimuli provided by the system to its environment. The set of reactions to be produced depends only on the set of input actions and the state before the transition. The set of reactions is also offered to a transfer mechanism, which transforms it into a, possibly empty, set of input stimuli.
The occurence of an input stimulus is called an event. An event thus consists of an action and a time stamp. An event is said to happen if the clock time equals the time stamp of the event.
The system maintains an agenda of events. To this agenda events can be added by the environment and by the system itself (namely by means of the transfer mechanism).

The last possibility allows a system to move through its state space without any influence from the environment.
Transitions are executed by a transition mechanism. This transition mechanism constantly inspects the agenda and comes into operation as soon as there are events happening. The number of transitions in a finite interval of time will be finite. This is the reason for calling the systems we study discrete.
The framework of a discrete dynamic system appears to be a generic model for many real life systems, being able to capture all relevant aspects of these systems. We call a real life system to be modelled an object system.
The generic model will be presented, as a 7-tuple <S,M,A,R,T,I,E>. The knowledge about an object system is modeled as the specification of the seven distinct components. Usually the knowledge of a system will be incomplete, e.g. because one does not know the future events from the environment.
It is also often the case that one only has partial knowledge of the transition mechanism. For instance, one knows that given a set of actions and a particular state, the system will perform a transition to one of a set of states. However, one does not know exactly which one.
There may be several reasons for constructing conceptual models of object systems. Among these we find the next ones:

- to support the design of the object system,
- to support the design of an information system that can be used to monitor and/or control the object system.

By monitoring is meant: keeping track of the process of the system, i.e. the sequence of states. Controlling means that the information system generates events for the object system.
The conceptual model of a system captures all knowledge that is available and thought relevant. In dealing with the acquisition of knowledge we distinguish between structural knowledge and behavioural knowledge.

By structural knowledge is meant all knowledge of a generic kind, e.g. the transition mechanism.
The structural knowledge of an existing object system is usually gained through a process of observation, hypothesizing and verifying hypotheses. When dealing with the design of an information system, the acquisition of the structural knowledge is done during the requirements engineering phase.
By behavioural knowledge is meant all knowledge of a factual kind, e.g. the state of the system at some moment.
We take the position that the only observable behavioural knowledge is the sequence of output stimuli of the object system. All other behavioural knowledge is deduced from the structural knowledge and the observations. Therefore one has to make sure in the requirements engineering phase that the information, which the information system has to provide, is deducible from the knowledge that can be acquired. The design of the mechanisms for gathering observations and disseminating output messages is part of the next phase, the functional design of the information system.
In section 2 we define a discrete dynamic system formally. Here we also consider the aggregation of a set of discrete dynamic systems into a new one. Section 3 discusses the knowledge one could have of discrete dynamic systems. Section 4 discusses some typical classes of discrete dynamic systems. In section 5 we consider the expression of the knowledge of a dds using first order languages. Finally, in section 6, we present some examples.

2. Definition of discrete dynamic systems.

The time domain of the systems we consider will be denoted by $\mathbb{R}^+$, the set of non-negative real numbers.

Definition 2.1 :

A **discrete dynamic system** (dds) is a 7-tuple

$$\langle S, M, A, R, T, I, E \rangle$$

where

S: a set called the **state space.**

A: a set called the **action space.**

R: a set called the **reaction space.**

M: a pair of function-valued functions $\langle MT, MR \rangle$ called
the **motor** where
$MT \in \mathcal{P}^{\frown}(A) \rightarrow (S \rightarrow S)$ the **transition function,**
$MR \in \mathcal{P}(A) \rightarrow (S \rightarrow \mathcal{P}(R))$ the **response function.**

T: a function, $T \in \mathcal{P}(R) \rightarrow \mathcal{P}(A * \mathbb{R}^+)$, the **transfer function**
such that $\exists\, \varepsilon > o : \forall B \subset R : \forall \langle a,d \rangle \in T(B) : d > \varepsilon$,
where d is a delay time.

I: an element of S , called the **initial state.**

E: a subset of $A * \mathbb{R}^+$, such that $\forall\ t \in \mathbb{R}^+$:
$\# \{\langle a,t' \rangle \mid \langle a,t' \rangle \in E \wedge t'' \leq t\}$ is finite,
and $\forall \langle a,t \rangle \in E : t > 0$;
this set is called the **external event set.**

A mechanical appreciation of a dds is as follows.
From the environment actions are imposed upon the system. An action occuring at a particular moment is called an event. Note that there may be more events at the same time. At time 0 a dds is in its initial state and it will stay there until the first moment at which there is an external event.

Then the motor is activated and the system will move instantaneously to the new state. The output, which is a set of reactions, is sent to the environment, and the transfer function determines a set of new events called **feedback events.** Their time stamps are relative to the time of the transition, i.e. these feedback events are added to the agenda of events with a time stamp that is the sum of the actual time and the delay time. Then the system stays in its new state until the clock arrives at the next moment for which there are events, external or feedback events. Since the delay times of the feedback events are bounded from 0 and since in each finite time interval there are at most finitely many external events, a dds has in each time interval at most finitely many transitions.

Formally the **behaviour** of a dds is described as follows.

Definition 2.2 :

Let <S, M, A, R, T, I, E> be a dds.

$\tau \in \mathbb{N} \rightarrow \mathbb{R}^+$; $\mathbb{N}$ is the set of natural numbers ;
(Note: instead of τ (n) we will also write τ_n)

τ_n is the time-stamp belonging to the n-th time in the life of the dds that the dds is, or will be, activated.

$\alpha \in \mathbb{R}^+ \rightarrow P(A)$
α(t) is the set of actions, exercised upon the dds at time t.

$\phi \in \mathbb{R}^+ \rightarrow P \quad (A * \mathbb{R}^+)$
ϕ(t) is the **agenda** of events, both external events and feed-back events, to be fed to the dds after t.

$\sigma \in \mathbb{R}^+ \to S$

σ is called the **process** of the dds.

$\sigma(t)$ denotes the **state** of the dds at time t.

$\rho \in \mathbb{R}^+ \to \mathcal{P}(R)$

$\rho(t)$ is the set of reactions the dds gives to the environment at time t.

such that : $\tau_0 = 0,\ \alpha(\tau_0) = \emptyset,\ \phi(\tau_0) = E,\ \sigma(\tau_0) = I,\ \rho(\tau_0) = \emptyset$;

and for $n \in \mathbb{N}$:

$$\tau_{n+1} = \min\{ t \mid \exists a \in A : \langle a,t\rangle \in \phi(\tau_n) \} ;$$
$$\alpha(\tau_{n+1}) = \{a \mid \langle a, \tau_{n+1}\rangle \in \phi(\tau_n) \} ;$$
$$\phi(\tau_{n+1}) = \{\langle a,t\rangle \mid (t > \tau_{n+1} \wedge \langle a,t\rangle \quad \phi(\tau_n)) \vee (\exists d : \langle a,d\rangle \in T(\rho(\tau_{n+1})) \wedge \tau_{n+1} + d = t)\} ;$$
$$\sigma(\tau_{n+1}) = MT(\alpha(\tau_{n+1}))(\sigma(\tau_n)) ;$$
$$\rho(\tau_{n+1}) = MR(\alpha(\tau_{n+1}))(\sigma(\tau_n)) ;$$

and for $t \in \mathbb{R}^+$

$$\alpha(t) = \emptyset \text{ if } \neg(\exists n \in N : t = \tau_n) ;$$
$$\phi(t) = \phi(\tau_n) \text{ if } \tau_n \leqq t < \tau_{n+1} ;$$
$$\sigma(t) = \sigma(\tau_n) \text{ if } \tau_n \leqq t < \tau_{n+1} ;$$
$$\rho(t) = \emptyset \text{ if } \neg(\exists n \in N : t = \tau_n) ;$$

Up to now we considered only one dds interacting with its environment. Often we encounter several interacting dss's. The interactions are modeled as transfers of reactions from one dds to another. Of course there may be communication with the environments also. An example of interacting dds's is an object system with its information system.

It also appears often useful to decompose a large dds into several interacting dds's.

We will consider a construction to build an **aggregate** dds from a set of dds's and a communication function that describes the interactions.

Definition 2.3 :

Let F be a dds-valued function with a finite domain J.
Let F (j) be denoted by

$$\langle S_j, M_j, A_j, R_j, T_j, I_j, E_j \rangle$$

Let C be a function-valued function with domain J * J
such that $\forall \langle i,j \rangle \in J * J : C(i,j) \in \mathcal{P}(R_i) \to \mathcal{P}(A_j * \mathbb{R}^+)$
$\wedge\, C(i,j)(\emptyset) = \emptyset$; $C_{jj} = T_j$;
(Note: we will write C_{ij} instead of C (i,j)).

The aggregate dds with respect to F and C is
$\langle S, M, A, R, T, I, E \rangle$
such that:

$S = \{ s \mid s$ is a function on J and $\forall\, j \in J: s(j) \in S_j \}$

$A = \{ \langle a,j \rangle \mid j \in J \wedge a \in A_j \}$

$R = \{ \langle r,j \rangle \mid j \in J \wedge r \in R_j \}$

$I =$ a function on J and $\forall\, j \in J: I(j) = I_j$

$E = \{ \langle\langle a,j \rangle,t \rangle \mid j \in J \wedge \langle a,t \rangle \in E_j \}$

$M = \langle MT, MR \rangle$, such that if $B_j = \{a \mid \langle a,j \rangle \in B$ for all $B \subset A \}$,
then for all $s \in S$:
$MT(B)(s) = s'$ where s' is a function on J and
$\forall\, j \in J : s'(j) = MT_j(B_j)(s(j))$, and
$MR(B)(s) = \{ \langle r,j \rangle \mid j \in J \wedge r \in MR_j(B_j)(s(j)) \}$

T = a function, $T \in \mathcal{P}(R) \to \mathcal{P}(A * \mathbb{R}^+)$ such that
if $H_j = \{ r \mid \langle r,j \rangle \in H \}$
for all $H \subset R$, then
$T(H) = \{ \langle\langle a,j \rangle,t \rangle \mid j \in J \wedge \exists i \in J : \langle a, t \rangle \in C_{ij}(H_i) \}$

It is easy to verify that an aggregate dds is indeed a dds. The function C is called the **communication function.** It generates events with a time stamp that is relative to the time of the transition, i.e. these events are added to the respective agendas of events with a time stamp that is the sum of the actual time and the relative time (which must be considered as a, possibly very small, delay time).
If we want to decompose a dds we go the other way round, i.e. we first identify J and then we define F.
If we want to consider a component-dds in isolation then we have to know next to its external events also the events it will receive from the other components and consider the union of these sets as the external event set for the dds.
The behaviour functions (τ, α , ϕ , σ and ρ) of a component dds are denoted by indexing the function name with j. Thus, e.g. τ_j denotes the τ-function of component dds_j.
As a final remark we note that the framework is also applicable for a discrete time domain, i.e. $\mathbb{N}$, because $\mathbb{N} \subset \mathbb{R}^+$.

3. Knowledge of a discrete dynamic system

The structure and behaviour of a discrete dynamic system are conceptualized according to the SMARTIE-framework. The specifications of the components of <S, M, A, R, T, I, E> define the structure and behaviour of the dds. We call these specifications the **knowledge** we have about the dds.
Usually the specifications are incomplete, meaning that there is only partial knowledge of (components of) the dds. For example, the component E will rarely be known in practice.
We distinguish between two kinds of knowledge, structural knowledge and behavioural knowledge.
By **structural knowledge** we mean all knowledge of a generic kind. In accordance with definition 2.1 this is the specification of the components S, M, A, R and T.
All knowledge of a factual kind is called **behavioural knowledge.** In accordance with definition 2.2 this is the set of specifications of the functions α , ϕ , σ and ρ .
Note that the behavioural knowledge is completely determined by, and thus can be derived from, the structural knowledge and the specifications of I and E. In practice however, I and E are rarely known, and the only way of acquiring behavioural knowledge is by observing the behaviour of the dds.
We define the behavioural knowledge obtained by observing the behaviour of a dds up to time t as a function KOB with $\mathbb{R}^+$ as domain, and for each t $\in \mathbb{R}^+$:

$$\begin{aligned} &\mathrm{KOB}\ (t) = < \mathbb{A}\ (t),\ \Sigma\ (t),\ \mathbb{P}\ (t)> \text{, where} \\ &\mathbb{A}\ (t) \subset \{\alpha\ (q)\ \mid q < t\}\ , \\ &\Sigma\ (t) \subseteq \{\sigma\ (q)\ \mid q < t\}\ , \\ &\mathbb{P}\ (t) \subset \{\rho\ (q)\ \mid q < t\}\ . \end{aligned}$$

We take the position that only ρ is observable, and that α and σ may be deduced from the observed ρ together with the available structural knowledge. With regard to ϕ , we note that as far as the feedback events are concerned, ϕ may also be deduced from ρ and the available structural knowledge.

Structural knowledge can be acquired in two different ways. Either there is somehow a priori knowledge available (e.g. because the system is designed), or the structural knowledge is deduced from observed behaviour through a process of hypothesizing and verifying of hypotheses.
A system dds1 is said to be **structurally identical** to a system dds2 if the structural knowledge of dds1 equals the structural knowledge of dds2.
A system dds2 is said to be **behaviourally identical** to a system dds1 if $\rho_1 = \rho_2$.
A system dds2 is said to be **identical** to a system dds1 if it is both structurally and behaviourally identical to dds1.
Note, that to know that two systems are identical does not imply that there is complete knowledge of the components of
<S, M, A, R, T, I, E>.

A system dds2 that is identical to a system dds1 can be used as a **model** for the system dds1. This means that one can acquire knowledge about dds1 by observing the behaviour of dds2.
If the system dds2 consists of symbolic structures (symbolizing concepts that refer to a physical world), we call it a **conceptual system.** Consequently, dds2 is called a **conceptual model** of dds1. Suitable and well-known materialization media for symbolic structures are the human brains, digital electronic circuits and pencil and paper.
If dds2 is a conceptual system, and if it is structurally identical to a system dds1, then dds2 embodies the structural knowledge of dds1. A well-known use of systems of the type of dds2 can be found in discrete simulation. The system dds2 is then called a **simulation model** of dds1. Any behaviour of a system that is structurally identical to dds1 can be simulated through suited choices of the components I and E of dds2.

4. Some classes of discrete dynamic systems.

A dds is called a **set-valued-state-system** if there is some universe of elements U such that $S = \mathcal{P}(U)$

Most of the systems we consider are set-valued-state-systems.

A dds is called an **autonomous system** if it satisfies the next conditions :

1) $\exists\, a \in A : \quad E = \{\langle a , 1\rangle\}$;
2) $\forall\, B \subset R : \quad B \neq \emptyset \quad T(B) \neq \emptyset$;

A clock is a perfect example of an autonomous system. A specification will be given in section 6.

A dds is called a **slave system** if it satisfies the next conditions :

1) $A \in S$;
2) $\forall\, s \in S : \forall\, B \subset A : MT(B)(s) = s \,\Delta\, B$, where Δ is the symmetric set difference ;
3) $\forall\, H \subset R : T(H) = \emptyset$;

Thus a slave system is a system that only maintains a state, such that state maintenance consists of direct insertion and deletion of elements.

We consider a pair of systems dds1 and dds2. The system dds1 is any discrete dynamic system. The system dds2 is a conceptual system.

We call dds2 an **information system** of dds1, if the next condition is satisfied :

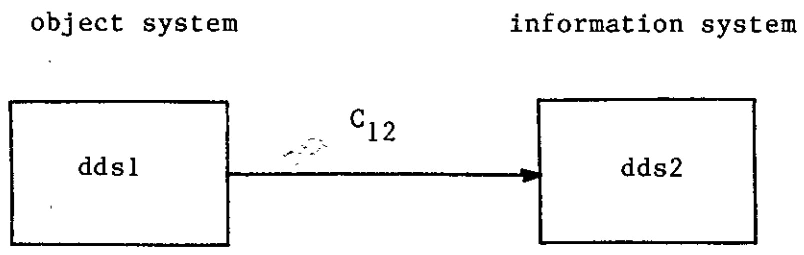
there is a correspondence function $\gamma : S_1 \rightarrow S_2$, and there is a delay function $\delta : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, such that $\forall\, t \in \mathbb{R}^+ : \gamma(\sigma_1(t)) = \sigma_2(t + \delta(t))$, and

$$\forall t_1, t_2 \in \mathbb{R}^+ : t_1 < t_2 \rightarrow (t_1 + \delta(t_1)) < (t_2 + \delta(t_2)) ;$$

The system dds2 usually is called the object system of dds1.

We distinguish between three kinds of information systems, named passive monitoring systems, active monitoring systems and control systems respectively.

An information system is called a **passive monitoring system** if the next conditions are met :



1) dds2 is a slave system ;
2) $\forall s \in S_1 : \forall B \subset A_1 : MR_1(B)(s) \supset (MT_1(B)(s) \Delta s)$;
3) $\forall t \in \mathbb{R}^+ : \forall H \subset R_1 : C_{12}(H) = \{\langle r, \delta(t)\rangle \mid r \in H\}$.

Condition 2 expresses that every state change of dds1 is made observable to dds2 by adding it to the response of dds1.
Condition 3 expresses that every such state change is directly supplied as input to dds2.
A passive monitoring system thus keeps track of the process of the object system. We say that dds2 is able to **postdict** the system dds1, by which we mean that dds2 can provide information about the past process of dds1. All registrative information systems are passive monitoring systems.
Dependent on the specification of δ, one may distinguish two special cases :

1) $\delta(t) = d + (t - \lambda_2(t))$,

where λ is an additional behaviour function, defined as :

$\lambda \in \mathbb{R}^+ \to \mathbb{R}^+$,

$\forall t \in \mathbb{R}^+ : \forall n \in \mathbb{N} : \tau_n \leqq t < \tau_{n+1} \rightarrow \lambda(t) = \tau_n$,

thus λ denotes the time stamp belonging to the most recent activation of the dds,

and

where d is a delay constant.

This case is called **batch monitoring** : only at particular moments $\tau_2(n)$ dds2 undergoes a state change, such that the new state corresponds to the last state of dds1, i.e. $\gamma(\sigma_1(\tau_2(n)) = \sigma_2(\tau_2(n) + d)$.

2) $\delta(t) = d$,

where d is a, usually small, constant.

This case is called **real-time monitoring.**

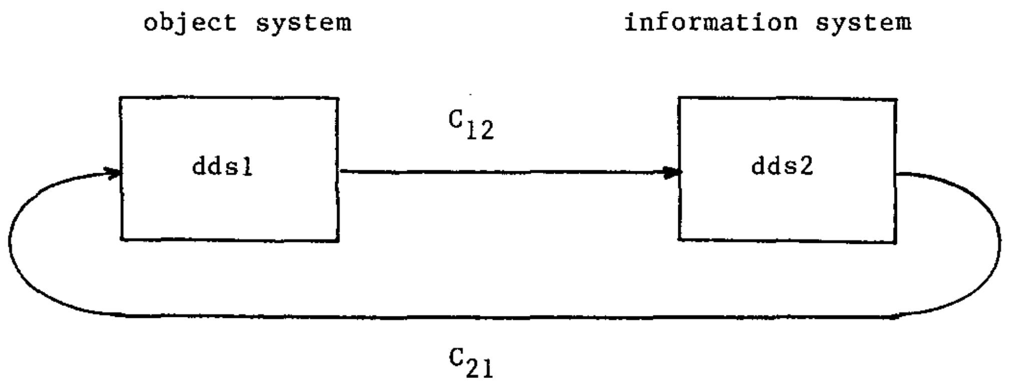
An information system is called an **active monitoring system** if the function C_{12} is specified as follows :

$$\forall s \in S_1 : \forall B \in A_1 : C_{12}(MR_1(B)(s)) = \{\langle a, \delta(t)\rangle \mid a \in B\}$$

expressing that copies of the actions imposed upon dds1 are passed to dds2.

Such a system is not only able to postdict the object system, but also to **predict it.** An information system that computes the trajectory of a heavenly body is an example of an active monitoring system.

A system dds2 is called a **control system** of a system dds1 if it is a monitoring system (either active or passive) and if C_{21} is specified as follows :



$\exists B \subset R_2 : C_{21}(B) \neq \emptyset.$

Example of control systems are production control systems, inventory control systems and airline reservations systems.

5. Knowledge expression

In this section a way of specifying the components of <S, M, A, R, T, I, E> is discussed.
Several disciplines deal with state spaces.
In control theory for instance, a state space is usually a linear space. In database theory a state space is expressed using a datamodel such as the relational model.
A promising approach to the representation of both S and M is offered by first order languages. In fact most data modeling languages can be considered as a restricted form of a first order language (fol) with some sugared syntax (cf. [Re82]).
We will consider here the expressive power of a fol for describing state spaces and transition mechanisms. We adopt a standard approach for defining fol`s from [ChLe73)] (see also [Ll84]), and start with a syntax definition.

Alphabet

- constants : U
- variables : X
- n-ary function symbols : F^n , n $\in$ {1,2,...}
- n-ary predicate symbols : P^n, n $\in$ {1,2,...}
- quantifiers : $\exists$, $\forall$
- logical operators : $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\neg$
- relational operators : $\leq$, $\geq$, $<$, $>$, $=$,
- arithmetic operators : + , − , * , div, mod
- punctuation symbols : (,) , : , ,

Terms

- constants and variables are terms;
- if $t_1,\ldots,$ t_n are terms and f $\in$ F^n then
  f $(t_1,\ldots,$ $t_n)$ is a term;
- if t_1 and t_2 are terms then
  (t_1+t_2), (t_1-t_2), (t_1*t_2), $(t_1 \mathrm{div} t_2)$, $(t_1 \mathrm{mod} t_2)$ are terms

Atoms

- the symbols Θ and Φ ;
- if $t_1, \ldots, t_n$ are terms and $p \in P^n$ then
  $p\ (t_1, \ldots, t_n)$ is an atom ;
- if t_1 and t_2 are terms then $(t_1 = t_2)$, $(t_1 \neq t_2)$
  $(t_1 < t_2)$, $(t_1\ \ t_2)$, $(t_1 > t_2)$, $(t_1 \geq t_2)$ are atoms

Formulas

- if Q is an atom then Q is a formula ;
- if Q and R are formulas then $(Q \wedge R)$, $(Q \vee R)$,
  $(\neg Q)$, $(Q \rightarrow R)$ and $(Q \leftrightarrow R)$ are formulas ;
- if Q is a formula and x is a variable then
  $(\forall\ x : Q)$ and $(\exists x : Q)$ are formulas.

For simplicity we assume that we restrict interpretations of a fol to cases where the domain of the interpretation is the set of integers. Furthermore, we assume that for all interpretations the truth values of Θ and Φ are true and false respectively. Interpretations can be defined as formulated in [ChLe73]. With this choice we have to represent objects in the real world by integers. Sometimes it is not useful to consider arithmetic operations on integers that are used as object identifiers, however it is syntactically correct. We will restrict our attention to interpretations where the relational and arithmetic operators have their usual meaning.
To demonstrate the power and weakness of a fol we give some examples.

Example 1.

Consider a state space described by the relational model. Suppose there are two relations P and Q with attribute sets {A,B} and {A,B,C} respectively. Further let {A} be a key for P and {A,C} for Q. Finally assume that referential integrity is required from Q to P according to attribute A, i.e. each A-value of a tuple in Q must occur in the set of A-values of all tuples in P.

We may express this state space using predicate symbols P, Q, A, B and C and some set of variables X including x, y, w, z and u. The scheme for this state space is expressed by the following clauses

- $\forall x : \forall y : \quad P(x,y) \rightarrow A(x) \wedge B(y)$

(note that this clause can be expressed by two Horn clauses)

- $\forall x : \forall y : \forall z : Q(x,y,z) \rightarrow A(x) \wedge B(y) \wedge C(z).$

Now the relations are defined. We continue with the key constraints.

$$\forall x: \forall y: \forall z: \; P(x,y) \wedge P(x,z) \rightarrow y=z$$

$$\forall x: \forall y: \forall w: \forall z: \; Q(x,y,z) \wedge Q(x,w,z,) \rightarrow y=w.$$

The referential integrity is defined by :

$$\forall x: \forall y: \forall z: \; P(x,y,z) \rightarrow \exists u: Q(x,u).$$

In Horn clauses we could formulate the last formula using a Skolem function symbol :

$$\forall x: \forall y: \forall z: \; P(x,y,z) \rightarrow Q(x, f(x,y,z)).$$

Example 2.

Suppose we want to express the summation of a function f over some set of elements of U satisfying a predicate p.
Informally this could be written as :

$$\sum_{z=x}^{y} \chi(p(z)) . f(z) \quad , \text{ where } \chi \text{ is the}$$

characteristic function, i.e. χ (true) = 1 and χ (false) =0.

We introduce in F^2 the function symbol Σ and we postulate:

$$\forall x: \forall y: \quad x > y \rightarrow \Sigma(x,y) = o$$
$$\wedge\ (x \leqq y \wedge p(x) \rightarrow \Sigma(x,y)=f(x)+\Sigma(x+1,y))$$
$$\wedge\ (x \leqq y \wedge \neg p(x) \rightarrow \Sigma(x,y)=\Sigma(x+1,y)).$$

If we restrict ourselves to first order languages then we have to define summation for each function f and each predicate p separately, in a similar way. However what we usually need is a generic function Σ that has two extra arguments : a function f and a predicate p.
This is only possible if we allow quantification over F_n and P_n ($n \in \{1,2,\ldots\}$). In that case we could postulate

$$\forall f \in F^1: \forall p \in P^1: \forall x \in U: \forall y \in U:$$
$$x > y \rightarrow \Sigma(f,p,x,y) = o$$
$$\wedge\ ((x \leq y \wedge p(x)) \rightarrow \Sigma(f,p,x,y)=f(x)+\Sigma(f,p,x+1,y))$$
$$\wedge\ ((x \leq y \wedge \neg p(x)) \rightarrow \Sigma(f,p,x,y)=\Sigma(f,p,x+1,y)).$$

We may simulate this in a first order language by introducing two extra variables i and j, a new function symbol $\tilde{f} \in F^2$ and a new predicate symbol $\tilde{p} \in P^2$. Further Σ has to be an element of F^4.
Then we define:

$$\forall i: \forall j: \forall x: \forall y:$$
$$x > y \rightarrow \Sigma(i,j,x,y)=o$$
$$\wedge\ ((x \leq y \wedge \tilde{p}(j,x)) \rightarrow \Sigma(i,j,x,y)=\tilde{f}(i,x)+\Sigma(i,j,x+1,y))$$
$$\wedge\ ((x \leqq y \wedge \neg\tilde{p}(j,x)) \rightarrow \Sigma(i,j,x,y)=\Sigma(i,j,x+1,y)).$$

If we want to use Σ for a special function f and a predicate p then we have to add for some values $u_1, u_2 \in U$:

$$\forall x: \ f(x) = \tilde{f}(u_1,x)$$
$$\forall x: \ p(x) = \tilde{p}(u_2,x).$$

Now we are allowed to use $\Sigma(u_1,u_2,a,b)$ in other formulas, where a and b are constants or variables.

To overcome the problems sketched above we will extend our language with a construct for summation by adding to the definition of terms :

for a variable x, a formula Q and a term t

$\Sigma\ x: Q : t$

is also a term.

The meaning of such a term is the integer value that is the sum of all values of t obtained by substituting for x a constant such that Q has the value true.

<u>Specification of S, A and R.</u>

A state space is characterized by a set of closed formulas [cf. Ll84]. This set is called the **scheme** of the state space. A **state** is a set of **ground atoms** i.e. predicate symbols with constants as arguments. Recall that the domain of interpretation is the set of integers. All the ground atoms in a state are given the value true, while all ground atoms not contained in the state are given the value false. This evaluation is called the closed world assumption [cf. Ll84, Re84]. All atoms of the form t_1 **r** t_2 where **r** is a relational operator and t_1 and t_2 are terms with constants substituted for variables are evaluated according to the rules of arithmetic.

Thus each state defines a truth evaluation for all ground atoms and hence for all closed formulas. If all formulas of a scheme evaluate to true for a given state, then the state is called **consistent** with the scheme. We use the notation s **cons** sch to express that the state s is consistent with scheme sch.

A state space S characterized by a scheme sch is defined by :

S = { s | s is a state and s **cons** sch}

An action space and a reaction space are described as sets of ground atoms from the same language. Often it is advisable to use different predicate symbols for the three spaces.

<u>Specification of M and T.</u>

We will consider a construction to define sets of ground atoms :

$\{ p(x_1,\dots,x_n) \mid g \mid Q \}$

where

p is a predicate symbol of P^n, $x_1,\dots,x_n$ are variables, g is a set of ground atoms and Q is a formula having at most $x_1,\dots,x_n$ as free variables.
It is the set of all ground atoms with predicate symbol p and constants $a_1,\dots, a_n$, such that the formula that is obtained by substituting $a_1,\dots,a_n$ for $x_1,\dots,x_n$ in Q evaluates to true with respect to the set of ground atoms g. Recall that we consider only interpretations where all ground atoms in g have the value true and all ground atoms not in g have the value false.
The general form of the definition of MT(B)(s) is :

$$\begin{aligned} MT(B)(s) &= t \quad \text{if } t \textbf{ cons } sch \\ &= s \quad \text{otherwise} \end{aligned}$$

where t is a finite union of sets of ground atoms constructed in the way decribed above with $g = s \cup B$.

For $s \in S$ and $B \subset A$ we define MR(B)(s) also as a finite union of sets of the form given above with $g = s \cup B$.

Finally T is constructed in a similar way by sets of the form :

$\{ \langle p(x_1,\dots, x_n), t \rangle \mid g \mid Q \}$

where only $x_1,\dots, x_n$ and t may occur as free variables in Q, and g is the argument of the function T.

6. Examples.

Example 1: Stepping robots

Consider a grid, defined as a set of grid points :

$\{ (x,y) \mid x \in \mathbb{N} \wedge y \in \mathbb{N} \wedge 0 \leq x < m \wedge 0 \leq y < n \}$

On this grid k robots are located ($k < m.n$).
No two robots can be located at the same point at any time.
Robots can move across the grid by making moves from one point to

another. If there are several moves for a robot to be performed at the same time these are accumulated into one new move. Moves can only be performed if the resulting placement of robots is feasible.
Every robot remembers its not performed moves and will get another chance to perform them.
The robots are triggered by external events to make moves and also by unperformed moves from the past.
This situation can be viewed as a dds. The structural knowledge is expressed below.

<u>**S**</u>

The fact that robot r is located at point (x,y) is expressed by the atom pos (r,x,y) where pos is a predicate symbol in P^3.
A state is a set of such atoms.
The scheme is defined by :

$$\forall r : \forall x : \forall y : \text{pos}(r,x,y) \rightarrow 1 \leqq r \leqq k \wedge 0 \leqq x < m \wedge 0 \leqq y < n$$

$$\forall v : \forall w : \forall x : \forall y : \text{pos}(v,x,y) \wedge \text{pos}(w,x,y) \rightarrow v = w$$

$$\forall r : \forall w : \forall x : \forall y : \forall z : \text{pos}(r,w,x) \wedge \text{pos}(r,y,z) \rightarrow w = y \wedge x = z$$

$$\forall r : \exists x : \exists y : 1 \leqq r \leqq k \rightarrow \text{pos}(r,x,y).$$

<u>**A**</u>

An elementary step is called a move and denoted by move (r,i,j) where r denotes a robot and i and j the movements in the directons of the first and second coordinate respectively.
Let A be defined by :

$$A = \{ \text{move}(r,i,j) \mid 1 \leqq r \leqq k \wedge i > 0 \quad j > 0 \}.$$

The steps will be made modulo m for the first coordinate and modulo n for the second one.

<u>**R**</u>

The reaction space is equal to the action space, i.e. R = A.

M

The motor of the dds, the pair <MT, MR>, is defined using the following abbreviations :

$$h\ (r) = \Sigma\ i : \Theta : (\ \Sigma\ j : \text{move}\ (r,i,j) : i\ )$$

and

$$v\ (r) = \Sigma\ i : \Theta : (\ \Sigma\ j : \text{move}\ (r,i,j) : j\ )$$

Note that h accumulates moves in the direction of the first coordinate and v in de second one.
Consider a set $B \subset A$ and a state s that is consistent with the scheme. Define a set of ground atoms t by :

$$t = \{\ \text{pos}\ (r,a,b)\ |\ s \cup B\ |\ 1 \leq r \leq k \wedge \exists\ x : \exists\ y : \exists\ w : \exists\ z : \text{pos}\ (r,x,y) \wedge h\ (r) = w \wedge v\ (r) = z \wedge a = (x+w) \bmod m \wedge b = (y+z) \bmod n\ \}.$$

If t and the scheme are consistent then :
$\quad MT(B)(s) = t$ and $MR(B)(s) = \emptyset$
else
$\quad MT(B)(s) = s$ and $MR(B)(s) = \{\text{move}\ (r,w,z)\ |\ w = h\ (r) \bmod m \wedge z = v\ (r) \bmod n \wedge w \neq 0 \wedge z \neq 0\}$

T

Finally we define T. We offer three possibilities.
First we consider the situation that there is no feedback; i.e $\forall H \subset R : T(H) = \emptyset$. Hence unperformed moves are discarded.
Secondly we consider a function T that produces feedback events for all not performed moves with the same time delay d, i.e.
$\quad T(H) = \{\ \langle\ \text{move}\ (r,w,z)\ ,\ d\ \rangle\ |\ H\ |\ \text{move}\ (r,w,z)\ \}.$
Note that only if there are other events occurring before these feedback events it is possible that these feedback events lead to performing moves.
Thirdly we consider a function T that produces feedback events as

was done in the second case, but now the time delays of these events will be different. The time delays assigned are proportional to the robot numbers, i.e.

$$T(H) = \{ < \text{move } (r,w,z) \text{ , } r.d > \mid H \mid \text{move } (r,w,z) \}$$

Note that this definition of T makes it possible that unperformed moves are performed at a future point in time without external events occurring in the mean time leading to robot moves.

Example 2: Clock

A simple example of an autonomous system, the behaviour of the arm indicating the seconds of a clock, is specified as follows :

S

The scheme is denoted by :

$$\forall\ x : \text{time } (x) \rightarrow 0 \leq x \wedge x < 60$$

$$\forall\ x : \text{time } (x) \wedge \text{time } (y) \rightarrow x = y.$$

Hence there is only one ground atom in each state that indicates time.

A

The action space consists of all ground atoms with predicate name step :

$$A = \{ \text{step } (x) \mid \Theta \}.$$

Each step (x) denotes a move forward of the arm with x seconds modulo 60.

R

The reaction space consists of only one reaction :

$$R = \{\text{done } (1) \}.$$

MT

For $B \subset A$ and $s \in S$:

$$MT(B)(s) = \{ \text{time } (x) \mid s \cup B \mid \exists\ y : \text{time } (y) \wedge x = ((\Sigma\ i : \text{step } (i) : i) + y) \bmod 60 \}.$$

Hence all steps are accumulated.

MR

For $B \subset A$ and $s \in S$:

$$MR(B)(s) = \{ \text{done } (1) \mid s \cup B \mid \text{step } (1) \}$$

if and only if step (1) was given the clock will give a reaction.

T

Every reaction of the type done (1) leads to a feedback event of action type step (1) precisely one second later :

$$T (\{ \text{done } (1) \}) = \{ < \text{step } (1) , 1 > \}$$

If we define the initial state to be: I = time (0), the clock may be started by any action of the type step (1).

References

[ChLe73]	Chang, C.L., Lee, R.C.T., Symbolic Logic and Mechanical Theorem Proving, Academic Press 1973.
[Ll84]	Lloyd, J.W., Foundations of Logic Programming, Springer-Verlag 1984.
[MyWo80]	Mylopoulos, J., Wong, H.K.T., Some features of the TAXIS model, sixt int. conf. on Very Large Data Bases, october 1980
[Re84]	Reiter, R., Towards a Logical Reconstruction of Relational Database, in : Brodie, M.L., Mylopoulos, J., Schmidt, J.W., (eds.) On Conceptual Modeling, Springer-Verlag, NY, 1984
[SoKu85]	Solvberg, A, Kung, C.H., On structural and behavioural modeling of reality, in : [StMe85]
[StHo85]	Studer, R., Horndasch, A., Modeling Static and Dynamic Aspects of Information Systems, in : [StMe85]

[StMe85] Steel jr., T.B., Meersman, R. (eds),
Proc. IFIP TC 2.6 WC on Database Semantics
(DS - 1),
North-Holland Publ., 1985

[VeFu85] Veloso, P.A.S., Furtado, A.L.,
Towards simpler and yet complete formal
specifications,
in :
Sernadas, A., Bubenko jr., J., Olivé, A. (eds),
Information Systems : Theoretical and Formal
Aspects, Elseviers Science Publ., Amsterdam, 1985

[Ze82] Zeigler, B.P.
System Theoretic Foundations of Modeling and Simulation
in : Elzas, M.S., Oeren, T.I., Zeigler B.P.,
(eds.) Simulation and Model based Methodologies :
an Integrative View, Nato ASI Series, Springer-
Verlag 1982.