

Recommendations on necessary concepts to model behavioural semantics

ICS 35.020

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW



Document Sharing Hub

Committees responsible for this Published Document

The preparation of this Published Document was entrusted to Technical Committee GEL/93, Design automation, upon which the following bodies were represented:

- British Aerospace plc
- Department of Trade and Industry
- Federation of the Electronics Industry
- Institution of Electrical Engineers
- Manchester Metropolitan University
- University of Manchester
- University of Southampton

This Published Document, having been prepared under the direction of the Electrotechnical Sector Board, was published under the authority of the Standards Board and comes into effect on 15 September 1996

© BSI 03-2000

The following BSI references relate to the work on this Published Document:
Committee reference GEL/93
Draft for comment 95/211311 SA

ISBN 0 580 26315 0

Amendments issued since publication

Amd. No.	Date	Comments

Contents

	Page
Committees responsible	Inside front cover
National foreword	ii
Recommendations on necessary concepts to model behavioural semantics	3

National foreword

This Published Document has been prepared by Technical Committee GEL/93 and is identical with CENELEC Report R117-007:1995 *Recommendations on necessary concepts to model behavioural semantics* published by the European Committee for Electrotechnical Standardization (CENELEC).

Summary of pages

This document comprises a front cover, an inside front cover, pages i and ii, pages 1 to 40 and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

October 1995

English version

Recommendations on necessary concepts to model behavioural semantics

This CENELEC Report has been prepared by the Technical Committee CENELEC TC 117, Electronic Design Automation (EDA). It was approved by TC 117 on 1994-10-25 and endorsed by the CENELEC Technical Board on 1994-12-06.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B-1050 Brussels

© 1995 Copyright reserved to CENELEC members

Contents

	Page
Appendix A Modelling Space and Time Phenomena	5
1 Introduction	5
2 The conception of information modelling	5
3 Syntactic versus semantic modelling	6
4 Finiteness and completeness	8
5 Modelling time	9
6 Modelling space-time phenomena	12
7 Related work	14
8 Conclusions	14
Figure 1 — Complete and incomplete figures	7
Figure 2 — The structure of a time point	10
Figure 3 — A linear time space	11
Appendix B Models of Time — Revised Report	16
1 Introduction	16
2 The basic time model	17
3 A measurable time model	21
4 An instantaneous event time model	23
5 Time properties	24
6 Event ordering predicates	25
7 Using time models	26
8 Time-continuous versus time-discrete behaviour	30
8.1 Modelling time-continuous behaviour	30
8.2 Modelling time-space phenomena	33
9 Time intervals	36
10 Conclusions	39
Figure 1 — The basic time space	18
Figure 2 — A linearised time space	20
Figure 3 — The VHDL modelling space	27
Figure 4 — Time continuous behaviour	32

Recommendations on Necessary Concepts to Model Behavioural Semantics

Gerhard Scholz,
Wolfgang Wilkes
University of Hagen
Germany

Hilary J. Kahn
University of Manchester
United Kingdom

The original assumption when starting the work on modelling VHDL was that EXPRESS would need to be extended because it had no intrinsic facilities for describing such concepts as state transitions, transactions or time. To investigate the requirements for modelling behaviour more fully, work was started on two topics:

- Existing methods and languages were reviewed to understand the capabilities they provide for modelling time and behaviour and to investigate whether their concepts are appropriate to the goal of modelling the behavioural aspects of VHDL. Four languages were examined: Milner's Calculus of Communicating Systems (MCCS), Actors, Petri Nets, and Temporal Logic. The result of this investigation is reported in [1].
- Work on modelling the behavioural aspects of VHDL using EXPRESS has been done. The result can be found in [2] – [6].

The results of both these approaches is briefly summarized as follows:

1) Result of the examination of existing methods:

It can be concluded from the examination that Petri Nets and Temporal Logic both provide features which are appropriate to the modelling of the behavioural semantics of VHDL. In contrast, MCCS and Actors provide rather fixed mechanisms. For example, in MCCS a fixed synchronous communication scheme is required; in Actors, an asynchronous communication mechanism is used. Thus, they can be used for describing the behaviour of a specific circuit being designed, but they are not suited to modelling the concepts which are necessary to describe the behaviour of a given circuit.

Petri Nets are based on the explicit introduction of *transitions*, which are defined by means of firing rules. An attempt to integrate this concept into EXPRESS would probably cause severe problems since it contradicts the "philosophy" of EXPRESS. EXPRESS does not provide any means of manipulating or changing the state of an object; it is mainly concerned with the description of states. Thus, the integration of concepts like transitions would change EXPRESS drastically.

Temporal Logic is based on some notion of time and event. However, it should be noted that there is no "standard" temporal logic. The normal way to capture behaviour in Temporal Logic is to describe the relationships between different states. This approach might seem to fit conceptually better into the approach of EXPRESS since it would not require the introduction of a new class of elements like *transitions* into EXPRESS. However, it is important to note that in Temporal Logic the concepts of time and time ordering are assumed axiomatically; this is not the case in EXPRESS. In EXPRESS time ordering operators must be explicitly defined.

2) Result of using EXPRESS to model VHDL behaviour:

This investigation shows that it is indeed possible to use EXPRESS to model VHDL behaviour. The general idea is to associate each event with a time point. Put another way, the solution is that a time point is associated with all the events which occur at that time point. This means that a time point has both a past set of events and a present set of events. It is this mechanism that allows basic time sequencing to be defined. By applying restrictions, it is possible to define different concepts of time e.g. with partial order or with total order. Time metrics may also be defined.

Thus, from a general perspective, EXPRESS can be thought of as having been "extended" in much the same way that temporal logic was introduced as an extension of predicate logic. In the case of Temporal Logic, new classes of variables were introduced into standard predicate logic to capture concepts such as **event** or **time point**, and for these new concepts specific predicates and functions were defined. In the same way, the time model in EXPRESS introduces some new entities which represent the concepts of **events** and **time points** and for these entities specific constraints, rules, and derived attributes are defined to capture the intended semantics. These newly defined entities can be used in applications (e.g. in a VHDL model) either as supertypes (thus getting more specialized events) or as attributes of other entities (e.g. to associate a static entity with a dynamic one).

In [2], different time models are presented. Starting with a very general “basic time model” based on **events** and **time points**, several extensions are presented including a “measurable time model” introducing **time values** and an “instantaneous event time model”, which allows the existence of several ordered events having the same time value and thus supports the modelling of delta delays. It has been shown that this model of time can be used to model the behavioural semantics of VHDL (see [3, 4, 5, 6]).

Considerable discussions arose about the question, whether these time models really provide a general frame for the modelling of behavioural and timing concepts. Although the models appeared to be well suited for the specific purpose of modelling VHDL which relies on a finite number of discrete simulation cycles, their usability for capturing time continuous behaviour was not very obvious. EXPRESS can deal with finite sets only, and that seemed to be a restriction for general purpose time models.

Finally it turned out, that in fact EXPRESS cannot model continuous time directly, but instead it can capture the continuous behaviour by describing directly any arbitrarily close finite approximation. This result has been integrated into the revised version of the time models which is attached as Appendix B. A similar approximation method is needed for spatial data when the infinite number of points which a surface in the Eukclidean plane consists of is described by a finite number of grid points. The paper of Appendix A “Modelling Space and Time Phenomena” focusses on these similarities and furthermore shows how these separate models for space and for time can be integrated for a common application.

Thus, this approach has not resulted in a model revealing deficiencies of EXPRESS. Instead, it has proved that time can be modelled in EXPRESS in the same way as may be done for “spatial” artifacts.

Thus, as a result of our efforts, our current view is the following:

- 1) There is no need to extend EXPRESS to define a behavioural model of VHDL.
- 2) A general “architecture” of a behavioural model of VHDL has been created.

The result of the development of a model of VHDL behaviour will be a specification of the underlying semantics of VHDL — but, of course, not an executable system. To create real implementations (e.g. to derive VHDL simulators from a model) might still require the transformation of EXPRESS models to an executable modelling language, for example, on a process modelling language such as GRAPES which is defined within EUROMETHOD.

In [7] it also shown that specific applications of an information model may well need additional specifications which go beyond the capabilities of EXPRESS. But these additions have to rely on the structures and restrictions as defined by the respective EXPRESS information models. The EXPRESS features have proven to be powerful enough to deal with complex behaviour. The addition of application specific information is beyond the scope and intent of EXPRESS.

The formal recommendation at this stage is, therefore, that EXPRESS does not need to be extended and that its capabilities are sufficient for the representation of VHDL behaviour.

References¹⁾

- [1] W. Müller, L. Schäfers, W. Wilkes: *Examination of Concepts in Existing Modelling Languages/Methodologies to model Behavioural Semantics*. ESPRIT Project 2972 ECIP2, Doc-No. ECIP2/HU/008-1 (Row 9), Oct. 92.
- [2] H.J. Kahn, C. Giumale: *Requirements for Modelling Behaviour and Timing*. ESPRIT Project 2972 ECIP2, Doc-No. ECIP2/UM/013-1 (Row 10), Oct. 92.
- [3] H.J. Kahn, C. Giumale: *Basic Model of VHDL*. ESPRIT Project 2972 ECIP2, Doc-No. ECIP2/UM/014-1 (Row 1), Oct. 92.
- [4] G. Scholz, W. Müller, H. Kahn, C. Giumale: *Review of the VHDL Basic Model and Impacts of the Model*. ESPRIT Project 2972 ECIP2, Doc-No. ECIP2/HU/016-1 (Row 98), July 93.
- [5] H.J. Kahn, C. Giumale: *An EXPRESS Information Model of VHDL 87*. ESPRIT Project 8370 ESIP, Doc-No. ESIP/UMAN/011.1, Dec. 94.
- [6] H.J. Kahn, C. Giumale: *Definition of an EXPRESS Information Model of VHDL 93*. ESPRIT Project 8370 ESIP, Doc-No. ESIP/UMAN/021.1, March 95.
- [7] J. Grebe, G. Scholz, W. Wilkes: *Possible Strategies to Derive a Procedural Interface Automatically from a Conceptual Model*, ESPRIT Project 2972 ECIP2, Document No. ECIP2/HU/017-1 (Row 20), July 1993.

¹⁾ NOTE all ECIP2 and ESIP Papers can be obtained from the authors.

Appendix A Modelling space and time phenomena

C.A. Giumale, G. Scholz, H.J. Kahn

1 Introduction

Within the electronic design (ECAD) community, several attempts have been started to capture the semantics of the objects of the domain by means of information models (IMs). For instance, the specifications of the Electronic Design Interchange Format [2] [13] and of the Design Representation Programming Interface of the CAD Framework Initiative [1] are accompanied by IMs. For the hardware description language VHDL [4], recently a detailed proposal of an IM of VHDLs salient features was provided [9].

All these IMs use the data specification language EXPRESS [3], a Draft International Standard of the ISO. EXPRESS is a structural object oriented information modelling language developed for the purpose of computer-interpretable representation and exchange of product data. It is restricted to the description of the structure of objects, it does not specify behaviour. At a first glance, this seemed to be a shortcoming for the description of geometric features as well as for the modelling of time dependent behaviour of ECAD objects. But it turned out, that the constraint mechanisms of EXPRESS have sufficient expressive power to capture these aspects.

This paper proposes a modelling method based on constrained sets. The geometrical shape of complex objects can be fully described using points sets belonging to a finite but arbitrary grid, their behaviour can be seen as a set of snapshots (events) taken at different moments in time throughout its evolution. Thereby the spatial extension of an object and its evolution in time are captured in a similar way, which allows an easy combination of both aspects. Behavioural properties can then be captured by sets of constraints attached to the space and time structure of the respective objects.

Chapter 2 introduces the underlying conception of information modelling. Chapter 3 addresses the question of “relevant information” a model should convey using a geometric example. Chapter 4 gives a proposal for a basic semantic model of geometry and discusses the associated finiteness problem. Chapter 5 presents a model of time based on events. Chapter 6 combines the separate models of the preceding chapters. The relationship of this presentation to current issues in the area of temporal, spatial, and spatiotemporal databases is discussed in chapter 7. Finally conclusions are drawn in chapter 8.

2 The Conception of Information Modelling

Given a universe of discourse D there are several levels to tackle a problem of D. At the conventional level the underlying concepts of D, their relationships and constraints, are implicitly used in order to write a program which eventually solves the problem. At a higher level the conceptual structure of D is explicitly represented and eventually used to generate programs for solving problems from D.

The first level is characteristic to computation whereas the second characterizes the field of information modelling. In the same way as a programming language is used to encode problem solving computations as a program, an information modelling language is used to represent the conceptual structure of a universe of discourse as an information model.

The main thrust of computation is to solve. The main effort in modelling is to describe. This goal is of significance in ECAD especially for supporting the transfer of design between different CAD systems and for design verification. As a result, interest in developing modelling languages and in modelling various CAD domains is growing, which — besides others — resulted in the above mentioned IMs of EDIF [2], CFI [1], and VHDL [9].

However, although the modelling activities have been successful, particularly in the field of EDIF, the IMs produced were intended for being used by humans for supporting standardization and to offer a clearer insight into the modelled domains. Tool support was restricted to graphical editors and simple checkers. There were no serious attempts so far at trying to use an IM directly as the controlling engine of an application program. This is presumably due to the fact that EXPRESS itself was under development until in 1992 a version was published which reached the status of a Draft International Standard.

The research concerning modelling in the field of data bases already proved that it is possible to step-wise convert a conceptual model of a data-base into an implementation of the data-base [11]. Research in the field of AI also proved that models of a universe of discourse can be used to solve problems in that universe [14]. It is expected that in future IMs will be used as well to drive programs which solve particular tasks in the domain described by the model. For example, the generation of procedural interfaces can be automated to some extent [10].

Nevertheless, in order to become operational a model has to contain relevant information to guide the problem solving activity and, in addition, this information must be available in a straightforward manner. There are still many discussions and different opinions of what is meant by “relevant information” and “straightforward manner”. This is particularly the case for the problem of how to build information models for complex domains which address time and space related behaviour. The following chapter discusses modelling styles with the example of a circle in a two-dimensional space.

3 Syntactic versus Semantic Modelling

Consider the very simple information model of a circle as given in [3], page 22.

```
ENTITY point;
  x      : REAL;
  y      : REAL;
END_ENTITY;

ENTITY circle;
  centre : point;
  radius : REAL;
DERIVE
  area   : REAL := pi*radius**2;
END_ENTITY;
```

The question to be asked is on which ground can this model be called a model of a circle ? The names chosen for the entities and their attributes cannot plead for this cause. In addition, what if the model is to be used in an application where one has to decide whether a given circle contains a given point or not ?

When looking at the model above, the representation of the concept of circle is rather syntactic. There can be several equivalent representations of the same kind. For instance, EDIF [2] describes a circle by two points, denoting the ends of the diameter. There is no obvious argument to prefer one or the other of these representations. This results mainly from the fact, that these models do not capture the semantics of a circle, but instead they contain information ready to be used provided that the computation package which uses it “knows” that it deals with a circle. The model above is rather a data structure encoding information about a circle but falls short from the main purpose of a true model: to explicitly describe the semantics of the described concept. The model above does not contain relevant information about the concept of a circle.

Consider now the model below, which sticks to the definition of a circle as given in geometry books: a set of points at a distance less or equal than the circle radius from the circle centre.

```
ENTITY point;
  x      : REAL;
  y      : REAL;
END_ENTITY;

ENTITY area;
  contents : SET OF point;
END_ENTITY;

ENTITY circle SUBTYPE OF ( area);
  centre   : point;
  radius   : REAL;
-- Attribute contents is inherited from entity area
DERIVE
  area     : REAL:= pi*radius**2;
END_ENTITY;
```

```

RULE points_in_circle( point, circle);
(* the contents of each circle consists of all points with distance to its centre less or
   equal to its radius *)
WHERE
  circle = QUERY( c <* circle | c.contents =
    QUERY(p <* point | distance(p,c.centre) <= c.radius ) );
END_RULE;

```

In the model above the relationship between circles and points is captured by a global rule which specifies the value for attribute *contents*. This model can be used to determine whether a given point P is in a circle C or not. The basic operation which has to be performed is to test if P IN C.contents.

In general, geometric elements as described above are areas in the Euclidean plane, which semantically are sets of points observing specific sets of constraints, as it is well known from mathematics. This point of view has considerable consequences for making models directly usable. For example, EDIF [2] defines geometric operations on “figures” and “figureGroups”. These operations can be described provided that their parameters are geometric elements modelled as sets of points. For example the difference operator can be represented as follows.

```

ENTITY area_difference SUBTYPE OF( area);
  Figure 1,
  Figure 2    : area;
DERIVE
  SELF\area.contents : SET OF point:=
    Figure 1.contents — Figure 2.contents;
END_ENTITY;

```

Other geometric operations may be more complex. But even this simple example can demonstrate the usefulness of the point set concept: The difference of e.g. two circles results in a more complex figure, which may be (among others) a donut. This model provides an easy semantic description of geometric operations and their results without the need to invent tricky parametric descriptions of arbitrarily complex figures.

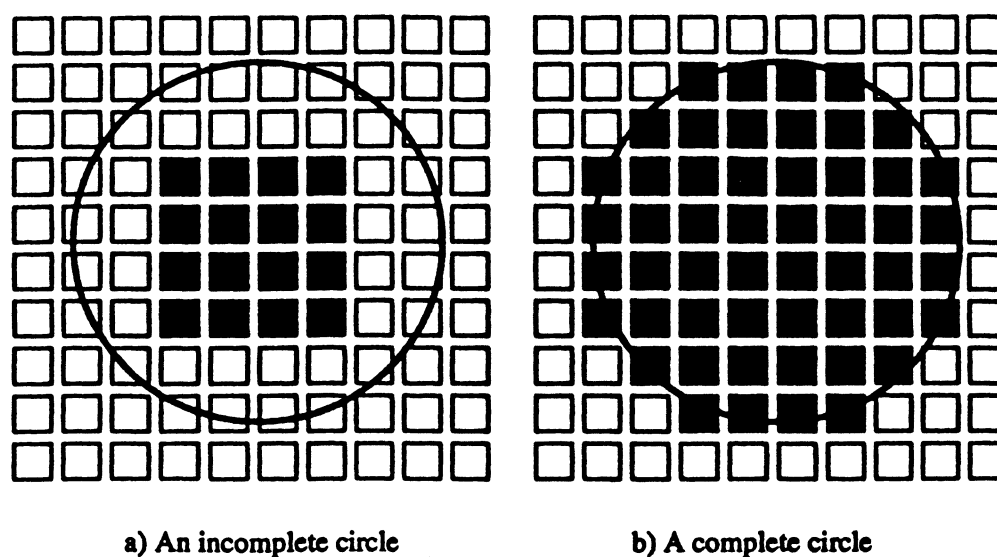


Figure 1 — Complete and incomplete figures

4 Finiteness and Completeness

The careful reader already noticed a problem concerning the representation of geometric elements as sets of points. From the mathematical point of view, point sets describing areas in the Euclidean plane are not only infinite, but not even countable. An EXPRESS entity on the other hand is the type description of an object, which eventually can be realized in an information base and therefore is finite. The rule *points_in_circle* does not consider all possible points, but just the finite number of points which are actually instantiated. The geometric object illustrated in part a) of Figure 1 is not a circle although — for a given set of *point* instances — it may satisfy rule *points_in_circle*. Obviously the circle is not complete. How can this problem be solved?

Two observations can be considered which lead from the conceptual infinity to a practical finiteness. First, areas in the ECAD domain do not have an arbitrary extension, but are limited. For example, in EDIF a geometrical shape is delimited by a “boundingBox”. Second, computers do not use real numbers in the mathematical sense, but only finite approximations. In EDIF, points are even represented with integer coordinates, related to a “grid” which needs to be defined. By these means we are half the way to solving the modelling of geometrical objects in EXPRESS.

The considerations above lead to the model below. An entity *universe* is defined in terms of *grid* and *bounding_box*. A global rule ensures, that for each set of instances according to the schema a unique *universe* exists. Each *geometry* is related to this *universe* by attribute *domain*. The contents of an *area* no longer refers to an arbitrary set of points, but to the well defined set of *universe.pixels* created by function *collect_points*.

```
ENTITY point;
  x      : REAL;
  y      : REAL;
END_ENTITY;

ENTITY_grid;
  x_unit : REAL;
  y_unit : REAL;
WHERE
  (x_unit > 0) AND (y_unit > 0);
END_ENTITY;

ENTITY_bounding_box;
  x_min : INTEGER;
  x_max : INTEGER;
  y_min : INTEGER;
  y_max : INTEGER;
WHERE
  (x_min < x_max) AND (y_min < y_max);
END_ENTITY;

ENTITY_universe;
  extension : bounding_box;
  precision : grid;
DERIVE
  pixels : SET OF point := collect_points( SELF);
END_ENTITY;

FUNCTION collect_points ( uni: universe): SET OF point;
LOCAL
  result : SET OF point := [];
END_LOCAL;
  REPEAT i := uni.extension.x_min TO uni.extension.x_max;
    REPEAT j := uni.extension.y_min TO uni.extension.y_max;
      result := result + point(i * uni.precision.x_unit,
                              j * uni.precision.y_unit);
    END_REPEAT;
  END_REPEAT;
  RETURN( result);
END_FUNCTION

RULE unique_universe FOR ( universe);
```

```

WHERE
  one_universe : SIZEOF( universe) = 1;
END_RULE;
ENTITY geometry ABSTRACT SUPERTYPE;
  domain      : universe;
END_ENTITY;

ENTITY area SUBTYPE OF ( geometry);
  contents    : SET OF point;
WHERE
  contents <= domain.pixels;
END_ENTITY;

ENTITY circle SUBTYPE OF ( area);
  centre      : point;
  radius      : REAL;
-- Attribute domain is inherited from entity geometry
DERIVE
  SELF\area.contents: SET OF point:=
    QUERY(p <= domain.pixels | distance( p, centre) <= radius );
END_ENTITY;

```

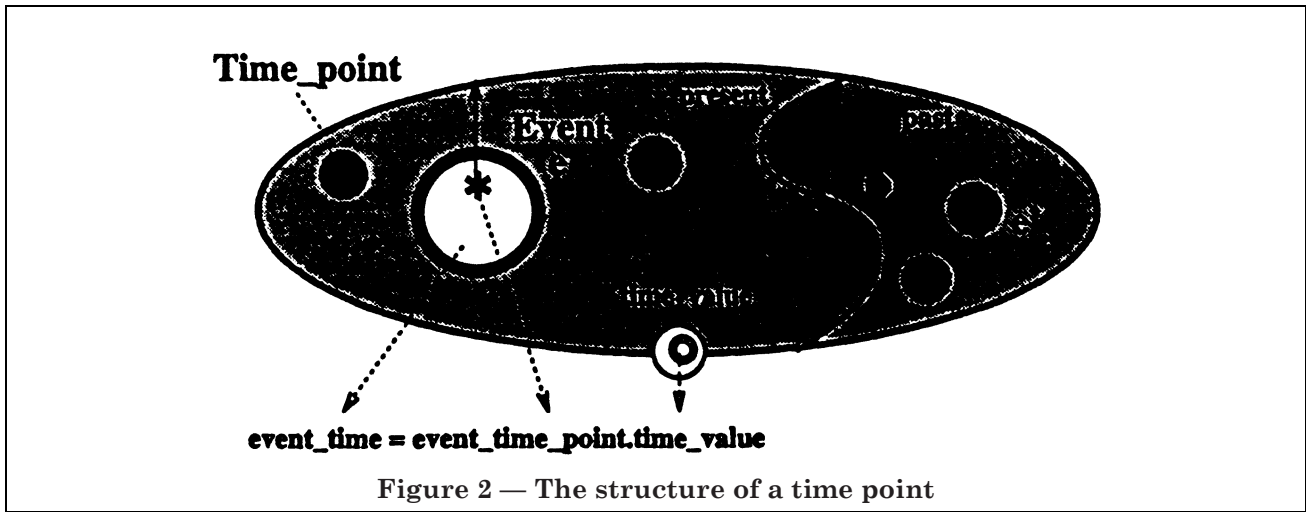
It should be noted that the representation of graphical primitives solely as finite sets of points is not sufficient. For instance, the property of a circle to be round can-not be expressed in this way. The “points” in part a) of Figure 1, depicted there as shaded squares, could similarly represent a polygon in the same universe. Therefore the “syntactic” attributes of the model of page 3 are still needed. But only their use in constraints or derived attribute constructs captures the semantics of a geometry which is beyond the pure result of a finite set of points.

Grid and *bounding_box* are unique in the scope of the model, but can be chosen arbitrarily. In order to decide for an arbitrary point P in the plane, whether it is inside a given circle C or not, the grid and bounding_box can be chosen in such a way that P is part of the pixel set of the universe. Then the test P IN C.contents can be performed. Thereby the model contains the full semantics of the circle, although it deals with finite sets only.

The geometry model above can be taken as a model of geometrical space. In the next chapter, it will be shown that time can be modelled in a similar way by considering it as a structured space of events instead of points. The geometric relationships will be replaced by temporal relationships.

5 Modelling Time

Time, as an abstract concept, is intended for event sequencing. It designates a space of time points reached from one another by “before” and “after”-like operators. For humans, time is also understood in terms of events. We remember events not time.



From this point of view it can be said that time has a corpuscular nature; quanta are events. Nevertheless, time is passing by, structuring the time point space in a way which makes it possible for an observer to orient in time by distinguishing the chronological ordering of events, regardless of the time distance between events. From this point of view time is continuous.

Considering events as fundamental and viewing time as a space of events comes near to the temporal logic proposed by Kowalski and Sergot [12]. However, the structure of this space is made explicit here rather than through being used to build logic formulae.

Time is seen as a space of time points. A particular time point (see Figure 2) contains the sets of events which are happening (the present events) or have happened (the past events). In this way the concepts “past” and “present” themselves can also be defined in terms of sets of events which can be ordered on a set inclusion basis.

Let Event be the generic set of events and Time_point be the set of time points. The following essential constraints have to be observed:

- Each $t \in \text{Time_point}$ contains two sets of events: $t.\text{present}$ and $t.\text{past}$. The present of a time point cannot be empty.
- Each $e \in \text{Event}$ is in the present of a unique time point ($e.\text{event_time_point}$.) It is said that e is created at $e.\text{event_time_point}$.
- For each $t \in \text{Time_point}$ and $e \in t.\text{past}$, all predecessors of e and all events concurrent with e (created at $e.\text{event_time_point}$) are in $t.\text{past}$.

These constraints are satisfied by the EXPRESS model below which uses the following set operators: IN for set membership, +, *, – for set union, intersection and difference, $\leq$ for set inclusion, = and $\neq$ for set equality and inequality, and $\square$ for representing the empty set.

```
ENTITY Time_value ABSTRACT SUPERTYPE;
END_ENTITY;
```

```
CONSTANT time_start: Time_value:=
    <application dependent constant>;
END_CONSTANT;
```

```
ENTITY Time_point;
    past      : SET OF Event;
    present   : SET [1: ?] OF Event;
    time      : Time_value;
WHERE
    r1: present * past =  $\square$ ;
    r2: is_equal( time, time_start) OR
        ( (past  $\neq \square$ ) AND is_greater( time, time_start) );
```