KIT REPORT 107

Inferences for Temporal Object Descriptions in a Terminological Representation System: Design and Implementation

---

Andreas Neuwirth

Technische Universität Berlin
Projekt KIT-BACK, FR 5–12
Franklinstr. 28/29, W–1000 Berlin 10

April 1993

Abstract

This paper describes design and implementation of an assertional component (ABox) of a temporal terminological representation system. With this ABox, and the temporal TBox developed by Martin Fischer (see [Fischer 92b]), it is possible to handle temporal object descriptions, and to infer 'new' (implicit) knowledge. An ABox query language offers a variety of possible queries to get certain information.
It is examined how existing algorithms for an ABox of a conventional terminological system can be transferred to the temporal system. Besides completely new algorithms have to be developed, because new problems arise with the temporal extension.

Contents

List of Figures

Chapter 1

Preface

Artificial intelligence is a young field of research, which exists no longer than 35 years. After little success in the beginning, the belief arose that essential improvements can be made by general approaches such as the General Problem Solver. But it appeared that this was wrong. Today it is assumed that problem specific knowledge is needed to solve non-trivial problems. The processing and representation of knowledge gets a bigger part in artificial intelligence research. We can distinguish several sorts of knowledge with corresponding inference mechanisms, for example uncertain, vague, non-monotonic or temporal knowledge.

One part of artificial intelligence is the terminological knowledge representation. Here we can differentiate between two formalisms: one for terminological (conceptual) knowledge and one for assertional knowledge (knowledge about objects).

In the so called 'TBox', the terminological knowledge is processed. Using concepts and roles, which correspond to single and binary predicates, problem specific terms can be defined. By using the term language, which consists of a small set of term forming operators, concepts and roles can be combined to form new terms. In this way a terminology can be constructed. In the TBox, nothing is said about particular objects, but knowledge of a special domain is gathered and represented.

The 'ABox' is for handling the assertional knowledge. With the help of the defined terminology, objects are described. A set of those descriptions build a world description. Using a suitable query language, special queries can be asked to the knowledge base. For the answers, not only the user asserted information is considered, but also the contained implicit knowledge, which is logical implied by the given knowledge. The possible ways of reasoning mechanisms depend on the used term language.

Well known agents of such representation systems are KL-ONE, KRYPTON, KANDOR, KL-TWO and BACK. KL-ONE is the first and most prominent of these systems. A good overview of KL-ONE can be found in [Brachman, Schmolze 85]. It was used in systems for understanding and generating natural language, interactive Information Retrieval etc. BACK was developed at the Technical University of Berlin by the KIT-BACK-group.

In the scope of a seminar, the μBACK-system was developed there too, which has a more experimental character. It is examined whether other forms of knowledge, like rules, defaults, probabilistic or temporal knowledge, can be integrated into the terminological disposition.

This report describes the integration of time into a terminological system. Based on the work of Martin Fischer (see [Fischer 91, Fischer 92a, Fischer 92b]) on a temporal TBox, the assertional component (ABox) of a temporal terminological system is described. Together

with his TBox, a complete temporal system was created. It was implemented in Prolog and can be used for experiments.

This disposition is based on works of Klaus Schild.[1] The non-temporal term language $\mathcal{NTF}^2$, which is used in μBACK, was extended by some operators from *tense logic*. This temporal language is used in the temporal μBACK, which is explained in this report.

It is not the first attempt to integrate time in a terminological system. Albrecht Schmiedel, for example, describes in [Schmiedel 88] and [Schmiedel 90] another way, which is totally different from this approach. It is based on the temporal logic of Yoav Shoham and the interval relations of James F. Allen.[3]

First, I will summarise some important aspects of the work of M. Fischer, because they are fundamental for this report. That is, for example, the temporal term language, or the so called 'timesets', which are needed to represent time (chapter 2).

In the following chapter, the temporal assertional language, and the temporal query language are described. With them it is possible to assert facts about objects into the knowledge base, and to ask for certain information. Afterwards it is shown how different object descriptions can be combined to a single one, and how the data structures to store the world description look like.

Chapter 5 and 6 are the centre of main effort, because the essential algorithms are presented there. At first, they are explained by examples and inference rules. Then the general course and structure of the implemented ABox is sketched, but the real Prolog code is not shown or commentated.

In the last chapter I will write some final remarks about the existing incompleteness, ways to increase the performance, and the advantages of restricted languages.

[1] see [Schild 91a] and [Schild 91b]
[2] see [Nebel 90]
[3] see [Shoham 87] and [Allen 83]

Chapter 2

Preliminaries

The most important work this report is based on, was written by Martin Fischer in his diploma thesis [Fischer 92a] and his report [Fischer 92b]. There he describes the language and the algorithms of his temporal TBox. In this chapter, I will recapitulate some of the main aspects which are necessary for the understanding of this report. First I will explain the terminological language of the TBox and the so called 'timesets' used for the representation of time. In the last section of this chapter, I will describe the TBox functions which are applied by the ABox. The combination of object descriptions (cf. [Fischer 91, p. 41]) will be described in section 4.1.

2.1 The Temporal Language $\mathcal{NTFT}$

The TBox is needed to build a terminology, which seems to be useful to model certain facts. A terminology consists of a set of terminological formulas such as

$\mathsf{term}_1 :< \mathsf{term}_2$ or $\mathsf{term}_1 := \mathsf{term}_2$.[1]

The terms are built by concepts and roles which are composed by the operators of the terminological language. Concepts correspond to single, and roles to binary predicates. The operators of the language determine the expressiveness of the language and the complexity of the required algorithms. The language $\mathcal{NTFT}$ was built on top of the non-temporal language $\mathcal{NTF}$[2] with additional operators from *tense logic*. Now I will explain the terminological language by giving an example. Thereafter you will find the syntax and semantics. A detailed explanation can be found in [Fischer 91].

creature	:<	**anything**	(T1)
human	:=	**alltime**(p_human **and** creature)	(T2)
woman	:=	human **and alltime**(p_woman **and not**(p_man))	(T3)
man	:=	human **and alltime**(p_man **and not**(p_woman))	(T4)

'Creature' is a primitive concept and used by the following concept definitions. The concept 'human' is defined by 'p_human' and 'creature'. For all p_-concepts it is true that p_NAME :< **anything**. They are useful to model primitive mutual exclusions or differences. A human is

[1] It is $\mathsf{term}_1 := \mathsf{term}_2 \stackrel{\text{def}}{=} \{\mathsf{term}_1 :< \mathsf{term}_2, \mathsf{term}_2 :< \mathsf{term}_1\}$

[2] see [Nebel 90, p. 55]

a creature with the additional property 'p_human'. This additional property is not explained, because in this model it doesn't matter. The operator **alltime** means, that if an object is an instance of the concept 'human' at any moment, it is instance of it at any time.

A woman is a human with the additional property 'p_woman' but without the property 'p_man'. The definition of 'man' is analogous to 'woman'. Therefore no object can be an instance of 'man' and 'woman' at the same time. Because the operator **alltime** is used, it is even excluded that an object can be a woman at one point of time and a man at another point.

born	:=	p_alive **and alltime**(**earlier**,**not**(p_alive))	(T5)
dead	:=	**alltime**(**now_or_later**,p_dead **and not**(p_alive))	(T6)
alive	:=	(p_alive **since nextt**(born)) **and** (p_alive **until** dead)	(T7)

If someone is born, he is 'p_alive' and alltime earlier not 'p_alive'. So nobody can be born twice. If someone is dead, he is analogously alltime later dead (not alive).

With the operators **since** and **until** it is possible to define valid periods in dependence of other concepts. If someone is instance of 'p_alive **since** born' at time point t, it means that he was 'born' at a former point t' and 'p_alive' at all time points t'' between t and t' ($t' < t'' \leq t$). So the validity of 'p_alive' begins one time point after 'born'. That is why the correct term is 'p_alive **since nextt**(born)', and **nextt** means the next time point. The definition of **until** is analogous to **since**. The term 'p_alive **until** dead' means now 'p_alive' is valid, and some time later 'dead' is valid, but nothing is said about the validity of 'p_alive', if 'dead' is valid. The exclusion of these two concepts is only because of the definition of 'dead'. So the concept 'alive' is valid in the period between 'born' and 'dead'.

full_age	:=	human **and alltime**([-18,0],alive)	(T8)
under_age	:=	human **and** alive **and sometime**([-17,0],born)	(T9)

A human is of full age, if he is at least 18 years alive. In the TBox, every period is a relative period. For example, if John is of full age at 1980, it implies he is alive the whole time (**alltime**) from $1980 - 18 = 1962$ to $1980 - 0 = 1980$. If someone is under age, then he was born some time in the last 17 years. So we are able to represent temporal certain (**alltime**) and uncertain (**sometime**) knowledge.[3]

spouse	:<	**anyrole**	(T10)
married	:=	**atleast**(1,spouse)	(T11)
unmarried	:=	**atmost**(0,spouse)	(T12)
monogamous	:=	**atmost**(1,spouse)	(T13)

'Spouse' is a role which is needed to model the marriage between two individuals. The operators **atleast** and **atmost**, taken from $\mathcal{NTF}$, are called number restrictions. They are required to restrict the number of role fillers by a lower or an upper bound. A role filler is the second object of a role relation.

Being married means, having (at least) one spouse, whereas unmarried means, having no spouse. A marriage is called monogamous, if there is at most one married partner.

[3] If I talk about uncertain knowledge in the following, I intend *temporal* uncertain knowledge, where you don't know *when* it is valid, but *that* it is valid.

$$
\begin{array}{rcl}
\langle formula \rangle & ::= & \langle term \rangle :< \langle term \rangle \\
\langle term \rangle & ::= & \langle concept \rangle \mid \langle role \rangle \\
\langle concept \rangle & ::= & \textbf{anything} \mid \textbf{nothing} \\
 & & \langle concept\text{-NAME} \rangle \mid \textbf{not}(\langle concept\text{-NAME} \rangle)^4 \mid \\
 & & \langle concept \rangle \ \textbf{and} \ \langle concept \rangle \mid \\
 & & \textbf{all}(\langle role \rangle, \langle concept \rangle) \mid \\
 & & \textbf{atleast}(\langle number \rangle, \langle role \rangle) \mid \textbf{atmost}(\langle number \rangle, \langle role \rangle) \mid \\
 & & \textbf{alltime}(\langle point\text{-}set \rangle, \langle concept \rangle) \mid \\
 & & \textbf{sometime}(\langle point\text{-}set \rangle, \langle concept \rangle) \mid \\
 & & \langle concept \rangle \ \textbf{since} \ \langle concept \rangle \mid \\
 & & \langle concept \rangle \ \textbf{until} \ \langle concept \rangle \mid \\
\langle role \rangle & ::= & \langle role\text{-NAME} \rangle \mid \textbf{anyrole} \\
\langle point\text{-}set \rangle & ::= & [\,] \mid \langle interval \rangle \mid [\langle interval \rangle | \langle point\text{-}set \rangle] \\
\langle interval \rangle & ::= & [\langle point \rangle, \langle point \rangle] \mid [-\infty, \langle point \rangle] \mid \\
 & & [\langle point \rangle, +\infty] \mid [-\infty, +\infty] \\
\langle point \rangle & ::= & \langle integer \rangle
\end{array}
$$

Figure 2.1: Syntax of the Temporal Language $\mathcal{NTFT}$

$$
\begin{array}{lcll}
\text{divorce} & := & \text{unmarried } \textbf{and} & \\
 & & \textbf{prevt}(\text{married } \textbf{and } \textbf{all}(\text{spouse}, \textbf{nextt}(\text{alive}))) & \text{(T14)} \\
\text{widowed} & := & \text{unmarried } \textbf{since} & \\
 & & (\text{married } \textbf{and } \textbf{all}(\text{spouse}, \textbf{nextt}(\text{dead}))) & \text{(T15)} \\
\text{divorced} & := & \text{unmarried } \textbf{since } \textbf{nextt}(\text{divorce}) & \text{(T16)}
\end{array}
$$

At the moment of divorce one is unmarried, and it is true at the previous point of time (**prevt**) that one is married and all spouses are still alive at the next time point (in contrast to 'widowed'). The operator **all**, also taken form $\mathcal{NTF}$, is called value restriction, requiring all fillers of a role (here 'spouse') to be instances of a given concept (here '**nextt**(alive)').

Now you should be able to understand the last two definitions. To get a deeper understanding of the syntax and semantics, look at figure 2.1 and definition 2.1.

Because there are no operators for building role terms, only the following three forms for terminological formulas are possible:

$$
\begin{array}{rcl}
\langle concept\text{-NAME} \rangle & := & \langle concept \rangle \\
\langle concept\text{-NAME} \rangle & :< & \textbf{anything} \\
\langle role\text{-NAME} \rangle & :< & \textbf{anyrole}
\end{array}
$$

Cyclic definitions, defining a concept directly or indirectly by itself, are not allowed.

[4] Only concept names with NAME :< **anything** are allowed.

Definition 2.1 (Semantics) Let $\mathcal{D}$ be a set and $\langle \mathcal{Z}, < \rangle$ be the time structure integer with the relation $<$. A structure $\mathcal{M}$ is an ordered triple $\langle \mathcal{D}, \mathcal{Z}, \mathcal{I} \rangle$. The interpretation function $\mathcal{I}$ is a function, mapping each object name o injectively to an element $\mathcal{I}[\mathsf{o}]$ of D, each concept c to a subset $\mathcal{I}[\mathsf{c}]$ of $\mathcal{D} \times \mathcal{Z}$ and each role r to a subset $\mathcal{I}[\mathsf{r}]$ of $\mathcal{D} \times \mathcal{D} \times \mathcal{Z}$. It is valid for concepts c, c_1, c_2, roles r, r_1, r_2, $d \in \mathcal{D}$, $t, t', t'' \in \mathcal{Z}$ and $T \subseteq \mathcal{Z}$:

$$
\begin{aligned}
\mathcal{I}[\textbf{anything}] &= \mathcal{D} \times \mathcal{Z} \\
\mathcal{I}[\textbf{nothing}] &= \emptyset \\
\mathcal{I}[\mathsf{c}_1 \textbf{ and } \mathsf{c}_2] &= \mathcal{I}[\mathsf{c}_1] \cap \mathcal{I}[\mathsf{c}_2] \\
\mathcal{I}[\textbf{not}(\mathsf{c})] &= (\mathcal{D} \times \mathcal{Z}) \setminus \mathcal{I}[\mathsf{c}] \\
\mathcal{I}[\textbf{atleast}(\mathsf{n}, \mathsf{r})] &= \{\langle d, t \rangle : |\{\langle d, d', t \rangle \in \mathcal{I}[\mathsf{r}]\}| \geq \mathsf{n}\} \\
\mathcal{I}[\textbf{atmost}(\mathsf{n}, \mathsf{r})] &= \{\langle d, t \rangle : |\{\langle d, d', t \rangle \in \mathcal{I}[\mathsf{r}]\}| \leq \mathsf{n}\} \\
\mathcal{I}[\textbf{all}(\mathsf{r}, \mathsf{c})] &= \{\langle d, t \rangle : \forall d' \, (\langle d, d', t \rangle \in \mathcal{I}[\mathsf{r}] \Rightarrow \langle d', t \rangle \in \mathcal{I}[\mathsf{c}])\} \\
\mathcal{I}[\textbf{alltime}(T, \mathsf{c})] &= \{\langle d, t \rangle : \forall t' \in T \, (\langle d, t + t' \rangle \in \mathcal{I}[\mathsf{c}])\} \\
\mathcal{I}[\textbf{sometime}(T, \mathsf{c})] &= \{\langle d, t \rangle : \exists t' \in T \, (\langle d, t + t' \rangle \in \mathcal{I}[\mathsf{c}])\} \\
\mathcal{I}[\mathsf{c}_1 \textbf{ since } \mathsf{c}_2] &= \{\langle d, t \rangle : \exists t' < t \, (\langle d, t' \rangle \in \mathcal{I}[\mathsf{c}_2]) \\
&\qquad \wedge \forall t'' \, (t' < t'' \leq t \Rightarrow \langle d, t'' \rangle \in \mathcal{I}[\mathsf{c}_1])\} \\
\mathcal{I}[\mathsf{c}_1 \textbf{ until } \mathsf{c}_2] &= \{\langle d, t \rangle : \exists t' > t \, (\langle d, t' \rangle \in \mathcal{I}[\mathsf{c}_2]) \\
&\qquad \wedge \forall t'' \, (t' > t'' \geq t \Rightarrow \langle d, t'' \rangle \in \mathcal{I}[\mathsf{c}_1])\} \\
\mathcal{I}[\textbf{anyrole}] &= \mathcal{D} \times \mathcal{D} \times \mathcal{Z}
\end{aligned}
$$

□

For better comprehension, the following abbreviations are possible in the TBox:

$$
\begin{aligned}
\textbf{nextt}(\mathsf{c}) &\stackrel{\text{def}}{=} \textbf{alltime}([1,1], \mathsf{c}) \\
\textbf{prevt}(\mathsf{c}) &\stackrel{\text{def}}{=} \textbf{alltime}([-1,-1], \mathsf{c}) \\
\textbf{alltime}(\mathsf{c}) &\stackrel{\text{def}}{=} \textbf{alltime}([-\infty,+\infty], \mathsf{c}) \\
\textbf{alltime}(t, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{alltime}([t,t], \mathsf{c}) \\
\textbf{alltime}(\textbf{later}, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{alltime}([1,+\infty], \mathsf{c}) \\
\textbf{alltime}(\textbf{now\_or\_later}, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{alltime}([0,+\infty], \mathsf{c}) \\
\textbf{alltime}(\textbf{earlier}, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{alltime}([-\infty,-1], \mathsf{c}) \\
\textbf{alltime}(\textbf{now\_or\_earlier}, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{alltime}([-\infty,0], \mathsf{c}) \\
\textbf{sometime}(\mathsf{c}) &\stackrel{\text{def}}{=} \textbf{sometime}([-\infty,+\infty], \mathsf{c}) \\
\textbf{sometime}(\textbf{later}, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{sometime}([1,+\infty], \mathsf{c}) \\
\textbf{sometime}(\textbf{now\_or\_later}, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{sometime}([0,+\infty], \mathsf{c}) \\
\textbf{sometime}(\textbf{earlier}, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{sometime}([-\infty,-1], \mathsf{c}) \\
\textbf{sometime}(\textbf{now\_or\_earlier}, \mathsf{c}) &\stackrel{\text{def}}{=} \textbf{sometime}([-\infty,0], \mathsf{c})
\end{aligned}
$$

2.2 Timesets

The timesets are needed for the representation of time. Time is described by a sequence of time points, and not by intervals as in [Allen 83]. Before and after each time point is an endless number of other time points, so it is suggested to describe time by the set $\mathcal{Z}$. The granularity, the "length of a time point", is freely choosable and depends on the used model. For a model of geology, you would choose another granularity as for a model of memory access in a computer.

For the calculation, it doesn't matter how time is presented for the user, by dates, by counting cycles or by another scheme. The only restriction is, that there has to be a function, mapping the scheme to $\mathcal{Z}$. Choosing the right mapping functions, even a combination of several schemes is possible. In our implemented system and in the examples in this report, no special conversion is done but plain $\mathcal{Z}$ is used. For a better demonstration you can imagine the numbers represent years with the granularity of one year.

A set of time points, a *pointset*, can be described by a usual set, i.e. {-2,-1,0,1,4,5,8}. But because these sets often contain sequences of time points and because there are infinite pointsets, they are represented by a list of closed intervals, i.e. [[-2,1],[4,5],[8,8]]. As declared in the syntax of $\mathcal{NTFT}$, single interval lists can be written with only one pair of brackets. It is true [[-1,2]] = [-1,2].

With timesets two sorts of temporal knowledge can be processed, certain and uncertain knowledge. Certain knowledge is valid over all time points (**alltime**) of a pointset. The expression a([-2,1]) contains all points from -2 to 1. The uncertain temporal knowledge is valid at least one time point of the pointset (**sometime**). So s([4,5]) means 4 *or* 5 *or* 4 and 5. The timesets are required for a unique representation of these two sorts of knowledge. Assume something is certainly valid over the periods [3,8] and [10,15], uncertain in [25,30] and some time after time point 49. This can be written as

{a([[3,8],[10,15]]),s([25,30]),s([50,+∞])}.

There are two minimality requirements for timesets. The first is, that no uncertain pointset may contain any other uncertain pointset of the same timeset. The set

{a([0,3]),s([10,20]),s([12,18])}

would not satisfy this requirement, because [12,18] is contained in [10,20]. The period s([10,20]) is superfluous and can be omitted, since s([12,18]) implies s([10,20]). Perhaps you think it will raise a loss of information, because s([10,20]) contains the additional possibilities of time points 10, 11, 19 and 20. But these points are not excluded by s([12,18]). It is only stated there is *at least one* valid time point in [12,18], but there can be any number of other valid time points.

The second requirement for timesets is that no *uncertain* pointset may contain any point of the *certain* pointset. Otherwise the uncertain pointset could also be omitted, because it is implied by the certain time point (see figure 2.2). Here is the definition of timesets we can find in [Fischer 92a, p. 9]:

Definition 2.2 (Timesets) Given the sets $T, T_1, T_2, \ldots \subseteq \mathcal{Z}$, a timeset $TS_{\mathcal{Z}}$ over $\mathcal{Z}$ is defined as the set $TS_{\mathcal{Z}} = \{\mathrm{a}(T), \mathrm{s}(T_1), \mathrm{s}(T_2), \ldots\}$, with all T_i, $i \geq 0$, satisfying the following two minimality requirements:

$\neg\exists\, T_j\,(T_j \subseteq T_i)$ and $T_i \cap T = \emptyset$ □

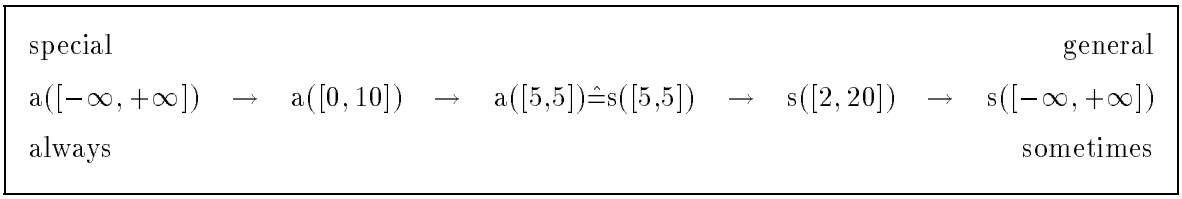


Figure 2.2: Relations between Timesets

For working with timesets, two operators $\cup_{\mathcal{TS}}$ and $\cap_{\mathcal{TS}}$ are defined, union and intersection. The union of timesets is the normal union of sets, with the additional check of the minimality requirements. So it can be necessary to omit some uncertain pointsets.

The intersection of timesets is a little bit more complicated. It is derived from the idea of coexistence. Suppose one fact is valid at TS, and an other fact is valid at TS'. Then these two facts are contemporary valid at $TS \cap_{\mathcal{TS}} TS'$ (see section 5.1.1). The contemporary time points of two *certain* timesets are calculated by normal set intersection:

$$\{\text{a}([0,10])\} \cap_{\mathcal{TS}} \{\text{a}([5,15])\} = \{\text{a}([5,10])\}$$

In an *uncertain* and a *certain* pointset, contemporary time points exist only if the uncertain pointset is wholly contained by the certain pointset:

$$\{\text{s}([5,10])\} \cap_{\mathcal{TS}} \{\text{a}([5,15])\} = \{\text{s}([5,10])\}.$$

The intersection of two uncertain pointsets is empty in every case, because there is no guarantee of the existence of two contemporary time points.

Definition 2.3 (Intersection of Timesets) Given the two timesets TS and TS', with $TS = \{\text{a}(T), \text{s}(T_1), \ldots, \text{s}(T_n)\}$ and $TS' = \{\text{a}(T'), \text{s}(T'_1), \ldots, \text{s}(T'_m)\}$. Then the intersection $\cap_{\mathcal{TS}}$ is defined as

$$TS \cap_{\mathcal{TS}} TS' \stackrel{\text{def}}{=} \{\text{a}(T \cap T')\} \cup \{\text{s}(T_i) | T_i \subseteq T'\} \cup \{\text{s}(T'_j) | T'_j \subseteq T\}$$

with $1 \leq i \leq n$ and $1 \leq j \leq m$. The minimality requirements must be satisfied. □

2.3 Used TBox Functions

The ABox uses some functions of the TBox to handle concept terms. In our system there are no role terms, because there are no role building operators. The concept terms are transformed into an internal TBox representation – the normal form (NF) – by a syntactical and semantical normalization and an incoherence check. With these normal forms, the TBox algorithms are more efficient or more complete than with the original terms. The ABox stores the object descriptions as normal forms and can simply use these algorithms. It needs essentially four classes of TBox functions to manipulate normal forms:

- Creating NFs:
  normalize transforms a syntactical correct concept term in its normal form.

- Changing NFs:
  unify unifies two normal forms to a new one.
  add_nf_timeset adds a NF to a second one, which is valid at a certain timeset.

- Comparing NFs:
  time_compare calculates the timeset in which a NF is valid, if an other one is valid at time point 0.
  time_disjoint calculates the timeset in which a NF is not valid, if an other one is valid at time point 0.
  compare tests, if a NF is valid every time another NF is valid, that is if a more general NF subsumes a more special one.

- Analysing NFs:
  subterm calculates all primitive concepts, all negated primitive concepts, all value restrictions and all number restrictions of a given NF.
  A NF is splitted into different components[5] which are separately accessible. *nf_until*, for example, allows the access to the until component.

Some of the described functions exists in several variants, so that not only normal forms are allowed as arguments, but also components of normal forms or concept terms.

[5] see [Fischer 92a, S. 14]

Chapter 3

The Assertional Language

In this chapter I will introduce the assertional language. It is the temporal object description language $\mathcal{ATLT}$ for asserting facts in the ABox and the operators of $\mathcal{AQLT}$ for asking queries. The object description language was taken from [Fischer 92a, Fischer 92b] without changes, but the query language was expanded with some additional operators. As in the previous chapter, I will introduce the language by giving some examples. At the end of the first section, some important notions are explained.

3.1 Introduction

The ABox is needed to build a world descripition that holds the information when objects or pairs of objects are instances of some concepts or roles. For the object descriptions you can use the previous defined terminology of the TBox. A world description is built by a sequence of assertional formulas such as:

$\mathsf{o} :: \mathsf{c}$ **at_all** T
$\mathsf{o} :: \mathsf{c}$ **at_some** T
$(\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r}$ **at_all** T
$(\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r}$ **at_some** T

The first two formulas are concept tells and the other two are role tells. Concept tells determine when an object o is instance of a concept term c, and a role tell states when two objects o_1 and o_2 are instances of a role. With **at_all** and **at_some** you can express temporal certain facts, which are valid during the whole pointset T, and uncertain facts, which are valid at some time in T.

For the introduction of the assertional language and in the following chapters, I will use the TBox terminology of the previous chapter. For that reason I show all the formulas of the terminology on the next page, but without explanation. It is extended with the two formulas (T17) and (T18), but they are not difficult to understand.

Now we come to the world description. At first there are two concept tells for the object Mary. The expression '... **at** t' is only an abbreviation for '... **at_all** [t,t]'.

Mary	::	born **and** woman **at** 1960	(A1)
Mary	::	(unmarried **since** **nextt**(born)) **and** alive **at** 1979	(A2)

creature	:<	**anything**	(T1)
human	:=	**alltime**(p_human **and** creature)	(T2)
woman	:=	human **and alltime**(p_woman **and not**(p_man))	(T3)
man	:=	human **and alltime**(p_man **and not**(p_woman))	(T4)
born	:=	p_alive **and alltime**(**earlier**,**not**(p_alive))	(T5)
dead	:=	**alltime**(**now_or_later**,p_dead **and not**(p_alive))	(T6)
alive	:=	(p_alive **since nextt**(born)) **and** (p_alive **until** dead)	(T7)
full_age	:=	human **and alltime**([-18,0],alive)	(T8)
under_age	:=	human **and** alive **and sometime**([-17,0],born)	(T9)
spouse	:<	**anyrole**	(T10)
married	:=	**atleast**(1,spouse)	(T11)
unmarried	:=	**atmost**(0,spouse)	(T12)
monogamous	:=	**atmost**(1,spouse)	(T13)
divorce	:=	unmarried **and prevt**(married **and all**(spouse, **nextt**(alive)))	(T14)
widowed	:=	unmarried **since** (married **and all**(spouse, **nextt**(dead)))	(T15)
divorced	:=	unmarried **since nextt**(divorce)	(T16)
working	:<	**anything**	(T17)
out_of_work	:=	**not**(working)	(T18)

The ABox describes a world with a woman Mary, born in 1960 and still alive in 1979. The fact that Mary is a woman in 1960 implies by definition (T3) that Mary is always a woman. So the answer of the question

when_is Mary **::** woman is {a([-∞,+∞])}.

The additional fact "Mary is a man in 1970" would be in conflict with (A1), and therefore it is rejected by the system. This conflict arises because of the definitions of 'man' and 'woman', which are mutually excluded. The declaration "Mary is born in 1965" would also be rejected, because it is excluded by definition that someone is born twice.

The query '**when_is** Mary **::** unmarried' yields the right answer {a([1960,1979])}. But why? It was said that Mary is an instance of 'unmarried **since nextt**(born)' in 1979, but nothing is said about the time between 1960 and 1979. Because of the definition of born, the concept 'born' can't be valid after 1960, and therefore 'unmarried' is valid from 1960 to 1979. This conclusion is done by the semantical normalization of the TBox, when all concepts to the object Mary are combined to the most specific description (see section 4.1). I will not describe such inferences, because this report is about ABox inferences, where more than one object is involved. Only by role tells objects come in relation to each other and it is necessary to make ABox inferences.

(Mary,John) :: spouse **at_all** [1980,1987] (A3)

Mary and John are married from 1980 to 1987. In this interval, Mary has a role filler of the role 'spouse', and the corresponding question

Mary **?** married **at_all** [1980,1987]

would be confirmed by the system. If we want to know which married couples exist between 1960 and 1985, we can ask the query

who_is spouse **at_some** [1960,1985].

The answer is {(Mary,John)}. These are all pairs of objects which are surely instances of the role 'spouse' in the given interval. There has to be at least either one certain time point or a whole uncertain pointset in the interval from 1960 to 1985 for being included in the answer.

The operator **who_is** is not only for roles, but also for concepts. It delivers the set of all objects which are instances of the concept at the given period. So it is the temporal variant of the **getall**-operator used in the μBACK- and the BACK-System which calculates all instances of a given concept.[1]

To discover which concepts are instantiated by an object in a given period we can use the operator **what_is**.

what_is	Mary **at** 1970	yields the set	{creature, human, woman, alive, under_age, monogamous, unmarried}.

All defined TBox concepts without the p_-concepts could be members of the answer. The first three concepts are simply implied by (A1). The validity of 'unmarried' (see above) implies 'monogamous', the reasons for 'alive' are similar to these of 'unmarried', and the validity of 'under_age' is easy to see.

⟨*world-description*⟩	::=	(⟨*object-tell*⟩ | ⟨*relation-tell*⟩)*
⟨*object-tell*⟩	::=	⟨*object*-NAME⟩ **::** ⟨*concept*⟩ **at_all** ⟨*point-set*⟩ | ⟨*object*-NAME⟩ **::** ⟨*concept*⟩ **at_some** ⟨*point-set*⟩
⟨*relation-tell*⟩	::=	(⟨*object*-NAME⟩,⟨*object*-NAME⟩) **::** ⟨*concept*⟩ **at_all** ⟨*point-set*⟩ | (⟨*object*-NAME⟩,⟨*object*-NAME⟩) **::** ⟨*concept*⟩ **at_some** ⟨*point-set*⟩

Figure 3.2: Syntax of the Temporal Assertional Language $\mathcal{ATLT}$

The syntax of the language for object descriptions is given in figure 3.2. It is assumed that different object names denote different objects. If not, we can't say anything about the number of role fillers at a specific time point. This condition is called *unique name assumption*. On the other hand it is *not* assumed that the world is fully described by the asserted facts (*closed world assumption*). Therefore the fact that Mary and John are married from 1980 to 1987 says nothing about the period before or after this interval. But we know that Mary was unmarried until 1979. So we can infer that they cannot be married before 1980.

Definition 3.1 (Model of a World Description) The structure $\mathcal{M} = \langle \mathcal{D}, \mathcal{Z}, \mathcal{I} \rangle$ is given as in definition 2.1. An assertional formula δ is satisfied by $\mathcal{M}$ under the following conditions:

$$
\begin{array}{lll}
\mathcal{M} \models \mathsf{o} :: \mathsf{c}\ \textbf{at\_all}\ T & \textit{iff} & \forall\, t \in T\ (\langle \mathcal{I}[\mathsf{o}], t\rangle \in \mathcal{I}[\mathsf{c}]) \\
\mathcal{M} \models \mathsf{o} :: \mathsf{c}\ \textbf{at\_some}\ T & \textit{iff} & \exists\, t \in T\ (\langle \mathcal{I}[\mathsf{o}], t\rangle \in \mathcal{I}[\mathsf{c}]) \\
\mathcal{M} \models (\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r}\ \textbf{at\_all}\ T & \textit{iff} & \forall\, t \in T\ (\langle \mathcal{I}[\mathsf{o}_1], \mathcal{I}[\mathsf{o}_2], t\rangle \in \mathcal{I}[\mathsf{r}]) \\
\mathcal{M} \models (\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r}\ \textbf{at\_some}\ T & \textit{iff} & \exists\, t \in T\ (\langle \mathcal{I}[\mathsf{o}_1], \mathcal{I}[\mathsf{o}_2], t\rangle \in \mathcal{I}[\mathsf{r}])
\end{array}
$$

A structure $\mathcal{M}$ is a model of a world description $\mathcal{W}$ with terminology $\mathcal{T}$, if it satisfies all formulas of $\mathcal{W}$ and $\mathcal{T}$. □

[1] see [Deumer, Neuwirth 91] and [Quantz, Kindermann 90]

If implicit knowledge of a world description is made explicit, 'new' facts are inferred. These facts are entailed by the world description. With the previous definition it is easy to say what entailment means.

Definition 3.2 (Entailment) Let $\mathcal{W}$ be a world description, $\mathcal{T}$ a terminology and δ any fact. The fact δ is entailed by $\mathcal{W}$ and $\mathcal{T}$, if all models of $\mathcal{W} \cup \mathcal{T}$ satisfy δ. □

Another important notion in the context of the ABox is the notion of inconsistency.

Definition 3.3 (Inconsistency) A world description $\mathcal{W}$ with terminology $\mathcal{T}$ is inconsistent, if there is no model for $\mathcal{W}$ and $\mathcal{T}$, that is when $\mathcal{W} \cup \mathcal{T}$ cannot be satisfied. □

Because any fact can be entailed by an inconsistent world description (see above), any ABox tell leading to inconsistency will be rejected by the system. Inconsistency arises when the most specific description of an object becomes incoherent by contradictory facts, that is when its interpretation is the empty set in all structures. But this means that the interpretation of the object is an element of the empty set and this is obviously impossible.

As in the TBox, there are abbreviations in the ABox too:

$$
\begin{array}{lcl}
\mathsf{o} :: \mathsf{c}\ \textbf{at}\ t & \stackrel{\text{def}}{=} & \mathsf{o} :: \mathsf{c}\ \textbf{at\_all}\ [t,t] \\
\mathsf{o} :: \mathsf{c}\ \textbf{at\_all\_times} & \stackrel{\text{def}}{=} & \mathsf{o} :: \mathsf{c}\ \textbf{at\_all}\ [-\infty,+\infty] \\
(\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\ \textbf{at}\ t & \stackrel{\text{def}}{=} & (\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\ \textbf{at\_all}\ [t,t] \\
(\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\ \textbf{at\_all\_times} & \stackrel{\text{def}}{=} & (\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\ \textbf{at\_all}\ [-\infty,+\infty]
\end{array}
$$

3.2 The Temporal ABox Query Language $\mathcal{AQLT}$

After forming a terminology and a world description, the user must have the possibility to get the stored information. For that reason the query language was created to get specific information from the ABox.

M. Fischer has defined a query language in [Fischer 92a] and [Fischer 92b] with four concept and four role queries. I have mentioned them in the previous section. With them, it is possible to ask specific questions, but it is only a small section of the range of all possible questions. The following scheme will show it. The queries are in the left column with variables O, O1, O2, C, R and TS representing the properties to be asked for. The meaning of these queries is described in the right column, and in the middle of these two the corresponding operator is declared, if it exists. Counting all combinations of possible queries will result in 8 concept and 16 role queries.

Mary **?** unmarried **at** {a([1976,1978])}		Is Mary unmarried from 1976 to 1978?
O **?** unmarried **at** {a([1976,1978])}	(**who_is**)	Who is unmarried from 1976 to 1978?
Mary **?** C **at** {a([1976,1978])}	(**what_is**)	What is Mary from 1976 to 1978?
Mary **?** unmarried **at** TS	(**when_is**)	When is Mary unmarried?
O **?** C **at** {a([1976,1978])}		What happened from 1976 to 1978?
O **?** unmarried **at** TS	(**instances_of**)	Instances of unmarried people
Mary **?** C **at** TS	(**concepts_of**)	Concepts of Mary
O **?** C **at** TS		Output of the concept KB
(Mary,John) **?** spouse **at** {a([1982,1982])}		Is John the spouse of Mary in 1982?
(O,John) **?** spouse **at** {a([1982,1982])}		Whose spouse is John in 1982?
(Mary,O) **?** spouse **at** {a([1982,1982])}		Who is the spouse of Mary in 1982?
(Mary,John) **?** R **at** {a([1982,1982])}	(**what_is**)	How is Mary related to John in 1982?
(Mary,John) **?** spouse **at** TS	(**when_is**)	When is John the spouse of Mary?
(O1,O2) **?** spouse **at** {a([1982,1982])}	(**who_is**)	Who are spouses in 1982?
(O,John) **?** R **at** {a([1982,1982])}		Who has a relation to John in 1982?
(O,John) **?** spouse **at** TS		Whose spouse is John at any time?
(Mary,O) **?** R **at** {a([1982,1982])}		With whom has Mary a relation in 1982?
(Mary,O) **?** spouse **at** TS		Who is the spouse of Mary at any time?
(Mary,John) **?** R **at** TS		Relations between Mary and John
(O1,O2) **?** R **at** {a([1982,1982])}		Which relations exists in 1982?
(O1,O2) **?** spouse **at** TS		History of spouses
(O,John) **?** R **at** TS		Who has a relation to John at any time
(Mary,O) **?** R **at** TS		With whom has Mary a relation at any time?
(O1,O2) **?** R **at** TS		Output of the role KB

The results of the queries with a variable C or TS can be restricted by giving a C_0 or a T_0 so that only those concepts or periods belong to the results which are subsumed by C_0 or $\{a(T_0)\}$. It is imaginable to restrict the objects too, by giving a special set of objects.

I have enhanced the query language by the way of the scheme above. Now there are more possibilities of queries, but there isn't a new operator for every new possibility. Because of the variables in the scheme and with the support of the backtracking of Prolog, it is possible to ask combined queries. If we want to know whether Mary was of full age when she was married with John, we can ask

(Mary,John) **?** spouse **at** TS, Mary **?** full_age **at** TS.[2]

This question will be answered with TS = {a([1980,1987])}. It is possible to restrict a result by the use of the operator **in**.

(O **in** [Mary,John]) **?** born **at** {s([1960,1970])}
Mary **?** (C **in** unmarried) **at** TS

The first question tests whether Mary or John is born at any time point between 1960 and 1970. The answer is 'O = Mary'. The result of the second query contains all TBox concepts, instantiated by Mary and subsumed by 'unmarried', with the belonging periods. In this example it is

C = unmarried, TS = {a([1960,1979])};
C = monogamous, TS = {a([1960,1979])}

Besides this new scheme, I have defined two operators for concept queries, **concepts_of** and **instances_of**, producing a chronological overview of concepts given an object, or objects given a concept. The command '**concepts_of** Mary' will produce a sort of a personal record.

a([-∞,1959])	:	[creature, human, woman]
a([1960,1960])	:	[creature, human, woman, born, alive, under_age, monogamous, unmarried]
a([1961,1977])	:	[creature, human, woman, alive, under_age, monogam, unmarried]
a([1978,1979])	:	[creature, human, woman, alive, full_age, monogam, unmarried]
a([1980,1987])	:	[creature, human, woman, married]
a([1988,+∞])	:	[creature, human, woman]
s([+∞,+∞])	:	[dead]

Perhaps it is a little bit surprising that the concept 'alive' is not shown in the interval from 1980 to 1987, but it was only stated that Mary is alive in 1979 (A2), after being born in 1960 (A1). Because defined roles are not allowed in the TBox, it is not possible to define that all instances of the role 'spouse' are living people.[3] Therefore, 'alive' is only valid until 1979. From 1960 to 1979, 'monogamous' is only valid because 'unmarried' is valid, hence it is not shown in later intervals.

The strange period s([+∞,+∞]) is the internal representation of the certain "time point" +∞. For (A2) and the definition of 'alive' (T7) the concept 'dead' is valid at s([1980,+∞]). Because the definition of 'dead' means from a certain time point and all time later you are dead, Mary is surely dead at time point +∞. The period s([1980,+∞]) is missing because of the second minimality requirement of timesets.[4]

The Syntax of the query language is shown in figure 3.3. Thereafter its semantics is described.

[2] In Prolog, all names beginning with a capital letter, are treated as variables, but here only O, C, R and TS are meant to be variables. But when working with the system, all object names must start with a lower-case letter.

[3] see [Quantz, Kindermann 90]

[4] see definition 2.2 page 7

$\langle$*object-ask*$\rangle$ ::= $\langle$*object*-NAME$\rangle$ **?** $\langle$*concept*$\rangle\langle$*at*$\rangle$ $\langle$*timepoint-set*$\rangle$ |
when_is $\langle$*object*-NAME$\rangle$ **::** $\langle$*concept*$\rangle$ |
who_is $\langle$*concept*$\rangle$ $\langle$*at*$\rangle$ $\langle$*timepoint-set*$\rangle$ |
what_is $\langle$*object*-NAME$\rangle$ $\langle$*at*$\rangle$ $\langle$*timepoint-set*$\rangle$ |
concepts_of $\langle$*object*-NAME$\rangle$ |
instances_of $\langle$*concept*$\rangle$ |
$\langle$*object-expression*$\rangle$ **?** $\langle$*concept-expression*$\rangle$ **at** $\langle$*timeset-expression*$\rangle$

$\langle$*relation-ask*$\rangle$ ::= $\langle$*relation*$\rangle$ **?** $\langle$*role*$\rangle\langle$*at*$\rangle$ $\langle$*abox-timepoint-set*$\rangle$ |
when_is $\langle$*relation*$\rangle$ **::** $\langle$*role*$\rangle$ |
who_is $\langle$*role*$\rangle$ $\langle$*at*$\rangle$ $\langle$*timepoint-set*$\rangle$ |
what_is $\langle$*relation*$\rangle$ $\langle$*at*$\rangle$ $\langle$*timepoint-set*$\rangle$ |
$\langle$*relation-expression*$\rangle$ **?** $\langle$*role-expression*$\rangle$ **at** $\langle$*timeset-expression*$\rangle$

$\langle$*object-expression*$\rangle$ ::= $\langle$*object*-NAME$\rangle$ | $\langle$*variable*$\rangle$ | $\langle$*variable*$\rangle$ **in** $\langle$*object-list*$\rangle$
$\langle$*concept-expression*$\rangle$ ::= $\langle$*concept*$\rangle$ | $\langle$*variable*$\rangle$ | $\langle$*variable*$\rangle$ **in** $\langle$*concept-list*$\rangle$ |
$\langle$*variable*$\rangle$ **in** $\langle$*concept*$\rangle$
$\langle$*relation-expression*$\rangle$ ::= $\langle$*relation*$\rangle$ | $\langle$*relation-var*$\rangle$ | $\langle$*relation-var*$\rangle$ **in** $\langle$*relation-list*$\rangle$
$\langle$*role-expression*$\rangle$ ::= $\langle$*role*$\rangle$ | $\langle$*variable*$\rangle$ | $\langle$*variable*$\rangle$ **in** $\langle$*role-list*$\rangle$
$\langle$*timeset-expression*$\rangle$ ::= $\langle$*timeset*$\rangle$ | $\langle$*variable*$\rangle$ | $\langle$*variable*$\rangle$ **in** $\langle$*point-set*$\rangle$

$\langle$*object-list*$\rangle$ ::= [ ] | [$\langle$*object*-NAME$\rangle$|$\langle$*object-list*$\rangle$]
$\langle$*concept-list*$\rangle$ ::= [ ] | [$\langle$*concept*-NAME$\rangle$|$\langle$*concept-list*$\rangle$]
$\langle$*relation-list*$\rangle$ ::= [ ] | [$\langle$*relation*$\rangle$|$\langle$*relation-list*$\rangle$]
$\langle$*role-list*$\rangle$ ::= [ ] | [$\langle$*role*-NAME$\rangle$|$\langle$*role-list*$\rangle$]

$\langle$*relation*$\rangle$ ::= ($\langle$*object*-NAME$\rangle$,$\langle$*object*-NAME$\rangle$)
$\langle$*relation-var*$\rangle$ ::= ($\langle$*object*-NAME$\rangle$,$\langle$*variable*$\rangle$) | ($\langle$*variable*$\rangle$,$\langle$*object*-NAME$\rangle$) |
($\langle$*variable*$\rangle$,$\langle$*variable*$\rangle$)
$\langle$*at*$\rangle$ ::= **at_all** | **at_some**

Figure 3.3: Syntax of the ABox Query Language $\mathcal{AQLT}$

Definition 3.4 (Semantics of the Operators when_is, who_is and what_is) Let $\mathcal{O}_\mathcal{W}$ be the set of all object names, existing in a world description $\mathcal{W}$, $\mathcal{NTFT}_\mathcal{C}$ the set of all concept terms and $\mathcal{C} \subseteq \mathcal{NTFT}_\mathcal{C}$ a subset of it. The semantics of the operators **when_is**, **who_is** and **what_is** for concepts c and roles r with regard to a terminology $\mathcal{T}$ and a world description $\mathcal{W}$ is defined as follows:

when_is : $\mathcal{O}_\mathcal{W} \times \mathcal{NTFT}_\mathcal{C} \rightarrow \mathcal{TS}$
a$(T) \in$ **when_is** o::c *iff* $T = \{t : \mathcal{T} \cup \mathcal{W} \models \mathsf{o} :: \mathsf{c}$ **at** $t\}$
s$(T') \in$ **when_is** o::c *iff* $\mathcal{T} \cup \mathcal{W} \models \mathsf{o} :: \mathsf{c}$ **at_some** T'
The minimality requirements for timesets are valid.

who_is$_{\langle at \rangle}$: $\mathcal{NTFT}_\mathcal{C} \times 2^\mathcal{Z} \rightarrow 2^{\mathcal{O}_\mathcal{W}}$
who_is $\mathsf{c}\langle at \rangle T = \{\mathsf{o} : \mathcal{T} \cup \mathcal{W} \models \mathsf{o} :: \mathsf{c}\ \langle at \rangle\ T\}$

what_is$_{\mathcal{C},\langle at \rangle}$: $\mathcal{O}_\mathcal{W} \times \mathcal{Z} \rightarrow 2^\mathcal{C}$
what_is$_{\mathcal{C},\langle at \rangle}$ $\mathsf{o}\ \langle at \rangle\ T = \{\mathsf{c} : \mathsf{c} \in \mathcal{C} \wedge \mathcal{T} \cup \mathcal{W} \models \mathsf{o} :: \mathsf{c}\ \langle at \rangle\ T\}$

when_is : $\mathcal{O}_\mathcal{W} \times \mathcal{O}_\mathcal{W} \times \mathcal{NTFT}_\mathcal{R} \rightarrow \mathcal{TS}$

$\mathrm{a}(T) \in$ **when_is** $(\mathsf{o}_1, \mathsf{o}_2)$::r *iff* $T = \{t : \mathcal{T} \cup \mathcal{W} \models (\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r}$ **at** $t\}$
$\mathrm{s}(T') \in$ **when_is** $(\mathsf{o}_1, \mathsf{o}_2)$::r *iff* $\mathcal{T} \cup \mathcal{W} \models (\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r}$ **at_some** T'
The minimality requirements for timesets are valid.

who_is$_{\langle at \rangle} : \mathcal{NTFT}_{\mathcal{R}} \times 2^{\mathcal{Z}} \rightarrow 2^{\mathcal{O}_{\mathcal{W}} \times \mathcal{O}_{\mathcal{W}}}$
who_is $\mathsf{c}\langle at \rangle T = \{(\mathsf{o}_1, \mathsf{o}_2) : \mathcal{T} \cup \mathcal{W} \models (\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r} \langle at \rangle T\}$

what_is$_{\langle at \rangle} : \mathcal{O}_{\mathcal{W}} \times \mathcal{O}_{\mathcal{W}} \times \mathcal{Z} \rightarrow 2^{\mathcal{R}}$
what_is$_{\langle at \rangle}$ $(\mathsf{o}_1, \mathsf{o}_2) \langle at \rangle T = \{\mathsf{r} : \mathcal{T} \cup \mathcal{W} \models (\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r} \langle at \rangle T\}$ □

In our implemented system, $\mathcal{C}$ is defined as the set of all TBox-concepts without the p_-concepts, but it is possible to modify this definition for special purposes.

By the operator **concepts_of** you get a chronological overview of an object. The result of such a query is a table with two columns. In the first column, there are the periods sorted in an ascending order, and in the second one are concepts which we would get by an **what_is**-query for these periods.

$$\begin{array}{lcl} \mathrm{TS}_1 & : & [\mathsf{c}_{11}, \mathsf{c}_{12}, \ldots] \\ \mathrm{TS}_2 & : & [\mathsf{c}_{21}, \mathsf{c}_{22}, \ldots] \\ \mathrm{TS}_3 & : & [\mathsf{c}_{31}, \mathsf{c}_{32}, \ldots] \\ & \vdots & \end{array}$$

The certain periods of the first column are simple intervals, not overlapping each other. Two certain intervals meeting each other have different concepts in the second column. Making a union of all periods containing a special concept in the second column, we will get the same period as by a corresponding **when_is**-query, that means the whole range of validity is shown.

With the aid of the operator **instances_of**, a chronological overview of a concept can be given. The resulting output is similar to that described above, but in the second row are *objects*, which are instances of the given concept in the corresponding period. The command '**instances_of** married' would show a table of all married people.

Now let's have a look at the new query scheme 'O **::** C **at** TS', at first at the simplest case of a concept query, where all three expressions are constants. It is tested, if it is entailed by the world description and the terminology that the object is instance of the given concept in the given period. The result is a 'Yes' or a 'No'.

If the object-expression is a variable, it will successively be substituted by all objects being an instance of the concept in the given period. The successive substitution is done by the backtracking of Prolog. If the concept-expression is a variable, it will successively substituted by all valid TBox concepts. The time variable is replaced by the corresponding timeset. In such a query more than one expression can be a variable. So in every step of backtracking, more than one substitution is necessary, but only those simultaneous substitutions are allowed which result in formulas entailed by the world description.

For restricting the answer set, that means the set of valid substitutions, the operator **in** was introduced. An object variable can be restricted by a given object set. Then it can only be substituted by objects of this set. For concept variables, two restrictions are possible, a restriction by a list of TBox concepts, or by a given superconcept. The superconcept restricts the substitutions to those concepts which are subsumed by it.

The scheme for role queries is defined in analogy.

As mentioned before, it is possible to ask combined queries with this scheme. By using the same variable, the result of one part of such a query can be used in other parts. All combinations of concept or role queries with or without restrictions are allowed.

When often working with restrictions by sets of objects or concepts, it is possible to enhance the system in a way that such sets can be separately predefined, and only the names of these sets need to be written in the queries.

Chapter 4

Data Structures

A world description contains explicit and implicit information about objects and relations between objects. These information has to be stored by the system. How the information is stored, especially the used data structures, is the subject of this section.

A simple method to store a world description is of course, to store it as given by the user. But there are some disadvantages. For every query, the whole knowledge base has to be examined to extract the necessary information. For using TBox-functions, normalization is required. These operations are done for every query, and for similar queries the same operations are done again and again. Therefore a good preparation and storage is necessary, to get a better performance.

My point of application is, as in BACK or μBACK[1], object centred, I try to collect all information about an object, not only explicit but also implicit information. An object is completely described by an intensional and extensional description. The intensional description is represented by the MSG (Most Specific Generalisation). It is a concept normal form which is composed of all user given and calculated concepts. How concept terms are combined is described in section 4.1. The extensional description is built by the role relations, connecting one object with other objects. The data structure for the intensional description looks as follows:

```
object(OName, Definitions, MSG)
```

For every object, name, user asserted facts, and its MSG is stored. The definitions are only stored for reconstructing the user asserted tells. So the intensional data structure is the same as in the normal μBACK, and it was easier to transfer the existing algorithms to the temporal system.

For the storage of the extensional description, I have considered more than one data structure:

1. `relation(RName, OName`$_1$`, OName`$_2$`, Timeset)`

   + simple format
   + Simple Prolog code for the calculation of all role fillers, inverse fillers, or all roles given two objects.

[1] see [Quantz, Kindermann 90] and [Deumer, Neuwirth 91]

- For the calculation of all role fillers, needed at several points in the program, expensive `setof`-operations have to be done.

2. `relation(OName, Fillers, InvFillers)`
   with `Fillers = [(`R_1`,[(`F_{11}`,`T_{11}`),..., (`F_{1n}`,`T_{1n}`)]),...,(`R_m`,[...])]`
   with `InvFillers = [(`R_1`,[(`O_{11}`,`T_{11}`),..., (`O_{1n}`,`T_{1n}`)]),...,(`R_m`,[...])]`

   + No more need of the `setof`-operator for the calculation of 'Fillers' and 'InvFillers'.
   + The extensional and intensional description can be combined in one structure: `object(OName, Defs, MSG, Fillers, InvFillers)`
   - need of more storage
   - The process of storing a role relation is a little bit more complicated than in the first data structure.

3. `relation(RName, [(`O_1`,RFList`$_1$`),...,(`O_n`,RFList`$_n$`)])`
   with `RFList`$_i$ `= [(`T_1`,[`O_{i11}`,...,`O_{i1m}`]),..., (`T_l`,[`O_{il1}`,...,`O_{ilm}`])]`

   + Atleast-abstraction (the counting of role fillers) is very easy.
   + Closed role filler sets are easy to find.
   - No simple calculation of all role fillers of an object.
   - The answer of role queries is more difficult.
   - need of more storage

After using the first structure in the beginning, I changed to the second one, so that the run time was a little bit shorter, depending on how much role relations and which Prolog version had been used. Using CProlog, the differences were more significant than with Quintus-Prolog. The third storage format was designed for reducing the time of atleast abstraction, but for the above-mentioned reasons it was not implemented.

4.1 Combination of Object Descriptions

In the normal μBACK all concept tells, not only user asserted but also system calculated facts, could be easy combined to one object description.

Mary :: woman and Mary :: unmarried $\Rightarrow$ Mary :: (woman **and** unmarried)

In the temporal system, it is not so effortless, because tells for different periods can't be directly combined.

Mary :: born **at** 1960 (4.1)

Mary :: unmarried **at** 1979 (4.2)

Only certain tells with identical intervals can be directly combined, but not uncertain facts with identical intervals. The facts 'o :: c_1 **at_some** T' and 'o :: c_2 **at_some** T' have another meaning as 'o :: c_1 **and** c_2 **at_some** T', because the last fact means that there is at least one time point at which both concepts are contemporary valid.

Lemma 4.1 Certain concept tells for same objects and same periods can be combined to one object description:

$$\mathsf{o} :: \mathsf{c}_1 \textbf{ at\_all } \mathsf{T} \quad \wedge \quad \mathsf{o} :: \mathsf{c}_2 \textbf{ at\_all } \mathsf{T} \quad \Rightarrow \quad \mathsf{o} :: \mathsf{c}_1 \textbf{ and } \mathsf{c}_2 \textbf{ at\_all } \mathsf{T} \qquad \square$$

Proof: Let $\mathcal{M}$ be any structure. It is true:

$$\begin{array}{ll} & \mathcal{M} \models \mathsf{o} :: \mathsf{c}_1 \textbf{ at\_all } \mathsf{T} \ \wedge \ \mathcal{M} \models \mathsf{o} :: \mathsf{c}_2 \textbf{ at\_all } \mathsf{T} \\ \textit{iff} & \forall\, t \in \mathsf{T}\ (\langle \mathcal{I}[\mathsf{o}], t\rangle \in \mathcal{I}[\mathsf{c}_1]) \ \wedge \ \forall\, t \in \mathsf{T}\ (\langle \mathcal{I}[\mathsf{o}], t\rangle \in \mathcal{I}[\mathsf{c}_2]) \\ \textit{iff} & \forall\, t \in \mathsf{T}\ (\langle \mathcal{I}[\mathsf{o}], t\rangle \in \mathcal{I}[\mathsf{c}_1] \ \wedge \ \langle \mathcal{I}[\mathsf{o}], t\rangle \in \mathcal{I}[\mathsf{c}_2]) \\ \textit{iff} & \forall\, t \in \mathsf{T}\ (\langle \mathcal{I}[\mathsf{o}], t\rangle \in (\mathcal{I}[\mathsf{c}_1] \cap \mathcal{I}[\mathsf{c}_2])) \\ \textit{iff} & \forall\, t \in \mathsf{T}\ (\langle \mathcal{I}[\mathsf{o}], t\rangle \in \mathcal{I}[\mathsf{c}_1 \textbf{ and } \mathsf{c}_2]) \\ \textit{iff} & \mathcal{M} \models \mathsf{o} :: \mathsf{c}_1 \textbf{ and } \mathsf{c}_2 \textbf{ at\_all } \mathsf{T} \end{array}$$

M. Fischer has shown that all concept tells can be transformed into that form, necessary for lemma 4.1. This is expressed by the following lemma:[2]

Lemma 4.2 All concept tells, not only certain but also uncertain, can be transformed in certain tells at time point 0:

$$\begin{array}{rcl} \mathsf{o} :: \mathsf{c} \textbf{ at\_all } \mathsf{T} & \Rightarrow & \mathsf{o} :: \textbf{alltime}(\mathsf{T},\mathsf{c}) \textbf{ at } 0 \\ \mathsf{o} :: \mathsf{c} \textbf{ at\_some } \mathsf{T} & \Rightarrow & \mathsf{o} :: \textbf{sometime}(\mathsf{T},\mathsf{c}) \textbf{ at } 0 \end{array} \qquad \square$$

These two lemmata lead directly to the following theorem about the combination of concept tells. The proof is trivial and therefore not described.

Theorem 4.1 (Combination of Concept Tells) All concept tells for an object, not only certain but also uncertain, can be combined to *one* object description by the proceedings given in lemma 4.1 and 4.2. $\square$

Now it is easy to see how the tells (4.1) and (4.2) can be combined.

Mary :: born **at** 1960 and Mary :: unmarried **at** 1979 is transformed to
Mary :: **alltime**(1960,born) **and alltime**(1979,unmarried) **at** 0.

As in the normal μBACK, the MSG can be stored as a concept normal form, describing the object at time point 0. Now it is possible for the ABox to use efficiently the implemented TBox algorithms, which need normal forms (see section 2.3). So all implications within one object description is done by the semantical normalization of the TBox. ABox inferences are only necessary, if more than one object is concerned with the entailment of a new fact.

[2] Proof see [Fischer 92b, P. 33]

Chapter 5

Algorithms

5.1 ABox Tells

All user asserted ABox tells for one object are combined into one object description, the MSG. Facts for one object may have consequences for other objects, which are related to it by role relations. When calculating the answer for a system query, this implicit knowledge has to be considered too. There is the possibility to calculate these consequences at query time (query-time-inferences) or at assert time (assert-time-inferences). Of course, a combination of these two ways is thinkable, where some inferences are precomputed and the rest is only calculated on demand. The precomputation has some advantages and disadvantages:

Assert-Time-Inferences	
Advantages	Disadvantages
shorter answer time	Creating a knowledge base needs more time.
Discovery of **Inconsistency** (contradictions) at assert-time.	need of more storage
No redundant calculation for similar or equal queries.	perhaps superfluous precomputation

In normal μBACK and in its temporal version, assert-time-inferences are preferred for reasons of efficiency. Another reason is the discovery of inconsistency at assert time. So mistakes in the used model can better be recognized and localized. assert-time-inferences are also preferred in other systems, for example [Quantz, Kindermann 90] and [Nebel 90, page 98].

The proceeding, where implicit knowledge is made explicit, I denote as completion. I distinguish the following four cases of completion:

- Forward propagation, where information is propagated from one object to its role-fillers,
- Backward propagation, which describes the reverse proceeding,

- Atleast-abstraction, where the role-fillers of one object are counted, and
- Specialisation of uncertain knowledge, where uncertain periods are reduced because of contradictions.

A special form of backward propagation is the abstraction of a closed role-filler-set. Since this is a procedure with difficult calculations, it is not precomputed, but only calculated on demand. For this reason it is only described in section 5.2.1.

Forward propagation, Atleast-abstraction, and abstraction of a closed role-filler-set exists in μBACK and BACK too. These algorithms were transferred to the temporal system. For the other forms of completion, completely new algorithms had to be created.

5.1.1 Forward Propagation

Forward propagation means that information is propagated from one object to its role-fillers. Information about role-fillers can be stated by value restrictions. For example, the fact that all of Mary's children are students, can be expressed by 'Mary **::** **all**(has_child,student)'. If we know that Robert is Mary's child, we can infer that he is a student.

Mary **::** **all**(has_child,student) and (Mary,Robert) **::** has_child $\Rightarrow$ Robert **::** student

If one of the two facts is asserted in addition to the other, the concept 'student' is propagated to 'Robert'. If it was not yet known that Robert is a student, and the propagation leads to a new, more special description of 'Robert', it has to be examined whether another new fact has to be propagated. This is necessary for example, if the concept 'student' is defined by another value restriction and a corresponding role relation exists. So a long chain of propagations can be caused by a single asserted fact. This chain is finished if all implicit knowledge is made explicit.

In temporal μBACK this proceeding is done too. The difference to the non-temporal forward propagation is that the valid period has also to be considered. The propagated term is only at those time points valid, where both facts surely coexist.

$$\mathsf{o}_1 :: \mathbf{all}(\mathsf{r},\mathsf{c})\ \mathbf{at\_all}\ \mathsf{T}_1 \quad \wedge \quad (\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\ \mathbf{at\_all}\ \mathsf{T}_2 \quad \Rightarrow \quad \mathsf{o}_2 :: \mathsf{c}\ \mathbf{at\_all}\ \mathsf{T}_1 \cap \mathsf{T}_2 \tag{5.1}$$

$$\mathsf{o}_1 :: \mathbf{all}(\mathsf{r},\mathsf{c})\ \mathbf{at\_some}\ \mathsf{T}_1 \quad \wedge \quad (\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\ \mathbf{at\_all}\ \mathsf{T}_2 \quad \overset{\mathsf{T}_1 \subseteq \mathsf{T}_2}{\Rightarrow} \quad \mathsf{o}_2 :: \mathsf{c}\ \mathbf{at\_some}\ \mathsf{T}_1 \tag{5.2}$$

$$\mathsf{o}_1 :: \mathbf{all}(\mathsf{r},\mathsf{c})\ \mathbf{at\_all}\ \mathsf{T}_1 \quad \wedge \quad (\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\ \mathbf{at\_some}\ \mathsf{T}_2 \quad \overset{\mathsf{T}_2 \subseteq \mathsf{T}_1}{\Rightarrow} \quad \mathsf{o}_2 :: \mathsf{c}\ \mathbf{at\_some}\ \mathsf{T}_2 \tag{5.3}$$

It is <u>not</u> true:

$$\mathsf{o}_1 :: \mathbf{all}(\mathsf{r},\mathsf{c})\ \mathbf{at\_some}\ \mathsf{T} \quad \wedge \quad (\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\ \mathbf{at\_some}\ \mathsf{T} \quad \Rightarrow \quad \mathsf{o}_2 :: \mathsf{c}\ \mathbf{at\_some}\ \mathsf{T}$$

Example:

Robert **::** **all**(examination,passed) **at_some** 1990
(Robert,mathematics) **::** examination **at_some** 1990

By the two facts *"There was* (at least) *one day*[1] *in 1990, where Robert has all examinations passed."* (this is especially true for all days where he has no examination taken), and *"At* (at least) *one day in 1990 Robert has taken an examination in mathematics"* can't be entailed that Robert has passed his examination in mathematics. If it is not sure that both facts coexist, no propagation can be done. □

Because of the coexistence of the facts, the rules (5.1)–(5.3) can be united to one single rule with the aid of the operator $\cap_{\mathcal{TS}}$.[2] The following rule explains the temporal forward propagation.

Rule 1 (Forward Propagation) Let $\mathcal{W}$ be a world description and $\mathcal{T}$ a terminology with the two relationships 'T_1= **when_is** o_1 **::** **all**(r,c)' and 'T_2= **when_is** (o_1,o_2) **::** r'. Then the fact 'o_2 **::** c' is valid in the period $\mathsf{T}_1 \cap_{\mathcal{TS}} \mathsf{T}_2$. □

[1] A granularity of one day is assumed
[2] see definition 2.3 page 8

Now let us come back to our terminology and world description of the sooner chapters and have a look at two examples of temporal forward propagations.

Mary	::	**all**(spouse,out_of_work) **at_all** [1970,1985]	(A4)
Mary	::	divorce **at** 1988	(A5)

These two ABox tells cause two forward propagations. From (A3) and (A4) and rule 1 can be inferred that 'John **::** out_of_work **at_all** [1980,1985]'. From (A3), (A5) and the definition of divorce (T14) can be entailed 'John **::** alive **at** 1988'. For all new ABox queries, these two entailed facts are also considered.

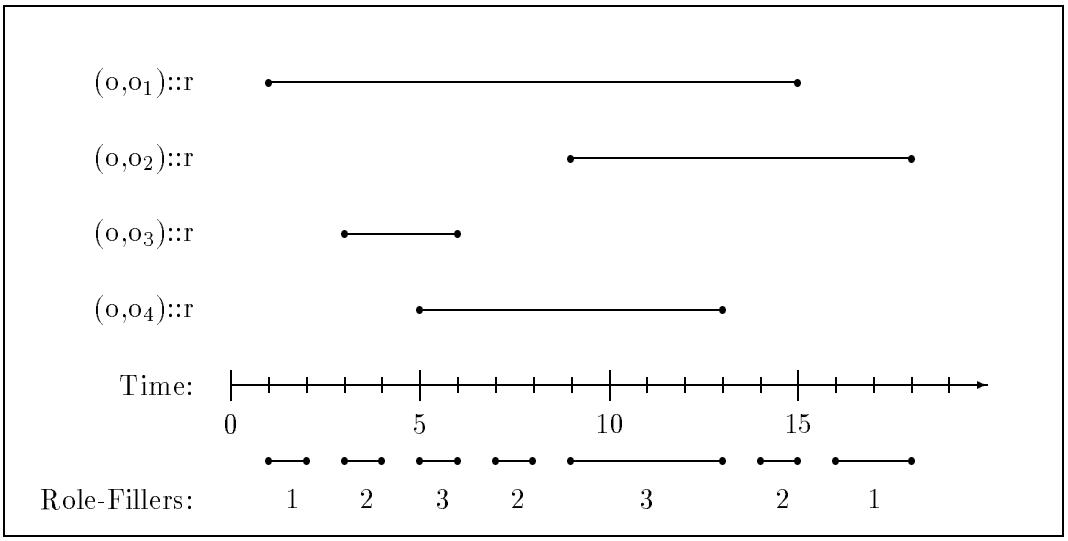


Figure 5.1: Calculation of the Number of Role-Fillers

5.1.2 Atleast-Abstraction

In normal μBACK there is another sort of implicit knowledge, that is knowledge about the number of role-fillers. The two facts

'(Mary,Robert) **::** has_child' and '(Mary,Ann) **::** has_child' imply
'Mary **::** **atleast**(2,has_child)'.

All role-fillers of an object and a particular role are counted, and this number is added to the object description as an Atleast-restriction.

In temporal μBACK this procedure is done too, but it is more complex. For a new role tell '$(\mathsf{o},\bar{\mathsf{o}})$ **::** r $\langle at \rangle \mathsf{T}$' T has to be intersected with all combinations of $\mathsf{T_i}$ of the existing role relations '$(\mathsf{o},\mathsf{o_i})$ **::** r $\langle at \rangle$ $\mathsf{T_i}$'. The result is the number of role-fillers in every intersected interval. For each of these intervals, an Atleast-restriction has to be abstracted for the object. In the worst case, the intersected intervals are twice as much as the role-fillers per role and object.[3]

Figure 5.1 shows an example with the following four role tells.

$$(\mathsf{o},\mathsf{o}_1) :: \mathsf{r}\ \textbf{at\_all}\ [1,15] \tag{5.4}$$
$$(\mathsf{o},\mathsf{o}_2) :: \mathsf{r}\ \textbf{at\_all}\ [9,18] \tag{5.5}$$
$$(\mathsf{o},\mathsf{o}_3) :: \mathsf{r}\ \textbf{at\_all}\ [3,6] \tag{5.6}$$
$$(\mathsf{o},\mathsf{o}_4) :: \mathsf{r}\ \textbf{at\_all}\ [5,13] \tag{5.7}$$

The hidden implicit knowledge contains seven Atleast-restrictions which have to be abstracted for the object o.

$\mathsf{o} :: \textbf{atleast}(1,\mathsf{r})\ \textbf{at\_all}\ [1,2]$ $\quad$ $\mathsf{o} :: \textbf{atleast}(2,\mathsf{r})\ \textbf{at\_all}\ [3,4]$
$\mathsf{o} :: \textbf{atleast}(3,\mathsf{r})\ \textbf{at\_all}\ [5,6]$ $\quad$ $\mathsf{o} :: \textbf{atleast}(2,\mathsf{r})\ \textbf{at\_all}\ [7,8]$
$\mathsf{o} :: \textbf{atleast}(3,\mathsf{r})\ \textbf{at\_all}\ [9,13]$ $\quad$ $\mathsf{o} :: \textbf{atleast}(2,\mathsf{r})\ \textbf{at\_all}\ [14,15]$
$\mathsf{o} :: \textbf{atleast}(1,\mathsf{r})\ \textbf{at\_all}\ [16,18]$

[3] If each of the n intervals is overlapping with any other interval, and if there are no contemporary beginning, ending or meeting (i.e. [2,5] and [6,8]) intervals, exact $2n-1$ intersected intervals exists.

Role relations, which are valid only at some time points (**at_some** T), need not to be considered at first. Only if this interval T is wholly outside of all other intervals or wholly inside of one of the calculated intersected intervals, then an Atleast-restriction for this period (**at_some** T) can be added to the object description. An additional role tell for the facts (5.4) - (5.7)

(o,o_5) :: r **at_some** [7,10] leads to the entailment

o :: **sometime**([7,8],**atleast**(3,r)) **or** **sometime**([9,10],**atleast**(4,r)) **at** 0

which can't be represented with our language.[4] Every 'compromise' like

o :: **atleast**$(3,\mathsf{r})$ **at_some** $[7,10]$ or

o :: **atleast**$(4,\mathsf{r})$ **at_some** $[9,10]$

is superfluous or wrong. The procedure of Atleast-abstraction can be expressed with the following rule:

Rule 2 (Atleast-Abstraction) Let $\mathcal{W}$ be a world description and $\mathcal{T}$ a terminology. The formula 'o :: **atleast**(n,r) $\langle at \rangle$T' can be entailed from them, if for n role-fillers is valid:

(o,o_i) :: r $\langle at \rangle \mathsf{T}_i$ and $\mathsf{T} = \mathsf{T}_1 \cap_{\mathcal{TS}} \mathsf{T}_2 \cap_{\mathcal{TS}} \ldots \cap_{\mathcal{TS}} \mathsf{T}_n$ □

Because of this rule, the query 'Mary **?** married **at_all** [1980,1987]' in section 3.1 was confirmed by the system. 'Married' is defined as '**atleast**(1,spouse)', and this has been abstracted for Mary, when the fact (A3) was asserted.

[4] We can see here the tight connexion of the operator **sometime** with the operator **or**, which is often avoided in other representation languages because of its complexity.

5.1.3 Specialisation of Uncertain Knowledge

This subsection is about a form of completion which is never needed in non-temporal systems, the reduction of uncertain periods. If an uncertain fact 'δ_s **at_some** T_s', no matter whether it is a concept or a role tell, is in contradiction with other facts at some time points, the period T_s can be reduced by these time points. If the result is only one possible time point, the uncertain knowledge is transformed into certain knowledge. If no possible time point remains, the world description is inconsistent.

The reduction of uncertain intervals for concept tells of *one* object is done by the semantic normalization of the TBox, when they are combined for the MSG.[5] But there are contradictions which are not within *one* object description, but which arise together with other descriptions. So ABox inferences are required.

Let's have a look at an example. Suppose we know that Mary had another spouse, but we don't really know when. Besides we know that she had not more than one spouse at the same time.

(Mary,Paul)	::	spouse **at_some** [1960,1990]	(A6)
Mary	::	monogamous **at_all** [1960,1992]	(A7)

If we ask when Mary and Paul are spouses, we will get the answer {s([1989,1990])}. Why? The role relation could not exist between 1960 and 1988, because Mary was unmarried from 1960 to 1979 (A2), married and monogamous from 1980 to 1987, and divorced in 1988 (A5).

In the following I will explain it a little more generally. The succeeding facts are contradictory, if they coexist, that is when the intersection of the three periods is not empty.

$$\mathsf{o}_1 :: \mathbf{all}(\mathsf{r}, \mathsf{c})\ \langle at\rangle \mathsf{T}_1 \tag{5.8}$$

$$(\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r}\ \langle at\rangle \mathsf{T}_2 \tag{5.9}$$

$$\mathsf{o}_2 :: \mathbf{not}(\mathsf{c})\ \langle at\rangle \mathsf{T}_3 \tag{5.10}$$

This contradiction is discovered by the propagations of c to o_2, because the MSG of o_2 becomes incoherent. It is similar to the following facts:

$$(\mathsf{o}, \mathsf{o}_1) :: \mathsf{r}\ \langle at\rangle \mathsf{T}_1 \tag{5.11}$$

$$(\mathsf{o}, \mathsf{o}_2) :: \mathsf{r}\ \langle at\rangle \mathsf{T}_2 \tag{5.12}$$

$$\mathsf{o} :: \mathbf{atmost}(1, \mathsf{r})\ \langle at\rangle \mathsf{T}_3 \tag{5.13}$$

The contradiction is discovered by counting the role-fillers and making an Atleast-abstraction. If the pointsets of the correlated facts are overlapping and one fact is not certain valid (**at_all** T_s), but uncertain valid (**at_some** T_s), then this pointset T_s can be reduced by the intersection of the other two pointsets (see figure 5.2).

The cases where the facts (5.10) or (5.13) are temporal uncertain, need not to be considered by the ABox, because it done by the forward propagation or Atleast-abstraction and the semantic normalization of the TBox.

In the following two rules for the specialisation of uncertain knowledge, the TBox function **time_disjoint**(c_1,c_2) is used. It calculates the timeset, when the concept c_1 is *not* valid, if c_2 valid.

[5] see [Fischer 92a, p. 20]

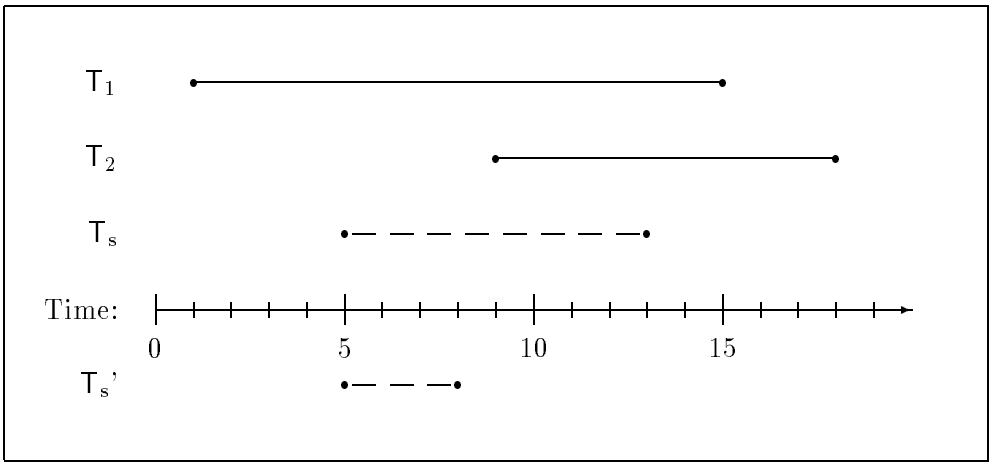


Figure 5.2: Reduction of Uncertain Intervals

Rule 3 (Specialisation of an Uncertain Value Restriction) Given the world description $\mathcal{W}$ with terminology $\mathcal{T}$ and the temporal uncertain fact 'o_1 **:: all**(r,c) **at_some** T_s'. If it is valid '(o_1,o_2) **:: r at_all** T_1' and a(T_2) $\subseteq$ **time_disjoint**(c,MSG$_{o2}$), T_s can be reduced by the intersection of T_1 and T_2:

o_1 **:: all**(r,c) **at_some** $T_s \setminus (T_1 \cap T_2)$ □

Rule 4 (Specialisation of an Uncertain Role Relation) Let $\mathcal{W}$ be a world description with terminology $\mathcal{T}$ and a temporal uncertain role relation '(o,$\bar{o}$) **:: r at_some** T_s'. A reduction of T_s is possible in the following two cases:

1. There is an object with '**o :: atmost**(n,r) **at_all** T' and n role fillers '(o,o_i) **:: r at_all** T_i'. Then at pointset $\bar{T} = T \cap T_1 \cap \ldots \cap T_n$ a closed role-filler-set exists. T_s can be reduced by $\bar{T}$, and it is valid:

   (o,$\bar{o}$) **:: r at_some** ($T_s \setminus \bar{T}$)

2. If it is valid 'o_1 **:: all**(r,c) **at_all** T_1' and a(T_2) $\subseteq$ **time_disjoint**(c,MSG$_{\bar{o}}$), T_s can be reduced by the intersection of T_1 and T_2:

   (o,$\bar{o}$) **:: r at_some** $T_s \setminus (T_1 \cap T_2)$ □

In the example given above, only rule 4.1 was used. In the period when 'unmarried', that is '**atmost**(0,spouse)', is valid, the rule was used in its simplest case, the case of the empty closed role-filler-set.

These two implemented rules satisfy the most existing ABox inferences for specialisation of uncertain knowledge. But a tighter examination shows that the number of facts, considered for the contradiction, can be very large, and it is not possible to include these cases with some basic rules. This incompleteness arises, when the value restriction doesn't conflict directly with the MSG of the role-filler, but it is only discovered after several "propagations". In these cases, the period of contradiction can not simply be calculated by the function **time_disjoint**. In opposite to the TBox, where all needed information is represented in *one* normal form, the complex relationship of all objects has to be considered. An algorithm realizing this, has to act in the following sense:

> For examination, every uncertain fact is asserted as a certain fact. If the system remains consistent, the uncertain interval has not to be reduced. But if it becomes inconsistent, the interval as to be reduced by those time points, which are in contradiction with the knowledge base.

Exactly this is the problem. If the description of any object becomes incoherent after several propagations and semantic normalizations caused by this 'test-tell', by which time points has the uncertain interval to be reduced? For finite pointsets, we could do such a 'test-tell' for every contained time point and see whether there is a contradiction or not. But without regard to the inefficiency of this proceeding, it is not usable for infinite intervals. It is one of the problems which remain unsolved.

Because such constellations of facts are presumably not very often, I think this incompleteness is not so critical. But it depends on the used model and should be heeded. For better comprehension, I will give a small example of this incompleteness.

o_1 :: **all**(r_1,**all**(r_2,c)) **at_some** [0,1]

(o_1,o_2) :: r_1 **at** 0	(o_1,o_4) :: r_1 **at** 1
(o_2,o_3) :: r_2 **at** 0	(o_4,o_5) :: r_2 **at** 1
o_3 :: **not**(c) **at** 0	o_5 :: **not**(c) **at** 1

The left facts contradict the possibility that the first fact is valid at time point 0, and the right facts contradict the time point 1. Therefore the world description is inconsistent, but this is not discovered. By rule 1 and rule 3 it is calculated that 'o_2 :: **all**(r_2,c) **at** 1' and 'o_4 :: **all**(r_2,c) **at** 0' are valid, but this is not in conflict with the first fact.

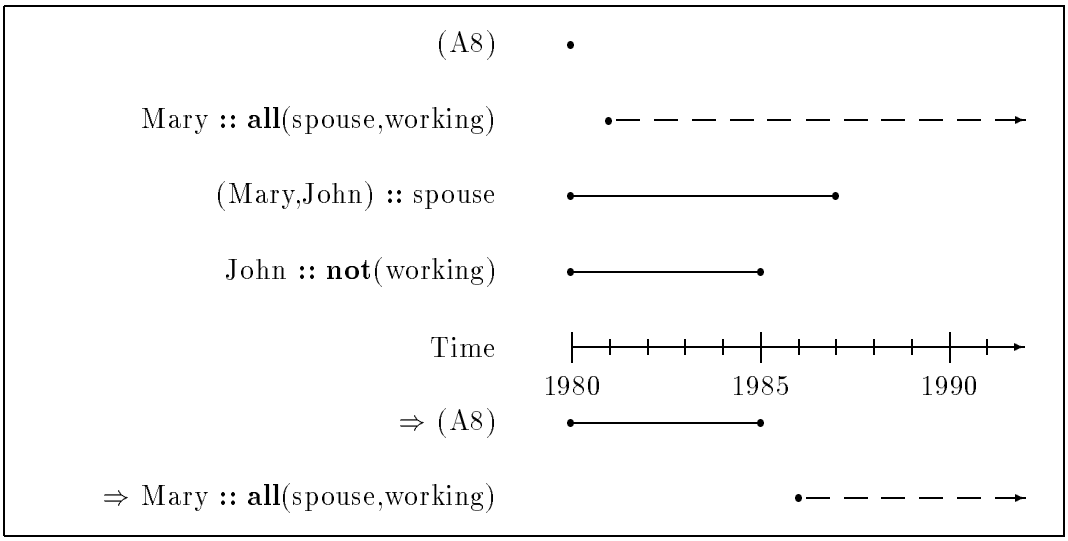


Figure 5.3: Example of a Backward Propagation

5.1.4 Backward Propagation

With backward propagations I mean all proceedings where the MSG of an object is influenced by its role-fillers. It contains the abstraction of closed role-filler-sets, described in section 5.2.1, and the elongation of the validity of **since-** and **until-**terms. Precisely rule 3 belongs also to the backward propagations, but I think it is more significant to put it together with the specialisation of uncertain role relations.

Now I will describe the circumstances, where information about role-fillers elongates the validity of **since-** and **until-**terms. This form of completion is, as the specialisation of uncertain knowledge too, not part of a normal terminological representation system. Let's have a look at the following example (consider figure 5.3):

At the marriage in 1980, Mary said that she really wants to work until her husband has found a new work.

Mary :: working **until** **all**(spouse,working) **at** 1980 (A8)

This formula means that Mary is working in 1980. She is working until some time later 'Mary :: **all**(spouse,work)' is valid, but nothing is said whether she is then working furthermore or not. If it is excluded that the spouse of Mary is working in 1981, (A8) has also to be valid in 1981. We know that John, the spouse of Mary, was out of work between 1980 and 1985. For that reason (A8) has to be valid from 1980 until 1985 too. If we ask the query

when_is Mary :: working,

we get the expected answer {a([1980,1985])}.

A little more generally, the following relationship is valid: If for an object o_1 is valid 'o_1 :: d_1 **until** d_2 **at** t_1' and it can be excluded that 'o_1 :: d_2' in the period a([t_1+1,t_2]), then 'o_1 :: d_1 **until** d_2 **at_all** [t_1,t_2]' is valid. For the operator **since** there are similar dependencies. Such inferences within one normal form are once more done by the semantic normalization of the TBox.

As in the previous section, we have to examine whether it is possible that some facts are valid at some time points. So the same problems arise. The dependencies, needed for the proof that a fact can't be valid at a fixed time point, can be in some cases very complex. I have decided to implement rules for the simple contradictions, these are the contradictions described by the formulas (5.8) to (5.13). As in the pervious section, I think that the resulting incompleteness is not so suspicious for the practicability of the system, but it has also to be considered.

Two questions have to be answered now. When are inferences needed to decide that o_1 can't be an instance of d_2 in the interval $[\mathsf{t}_1+1,\mathsf{t}_2]$? How do these inferences look like?

Looking at formulas (5.8) to (5.13), we can see that **until**-terms can only exist in (5.8), (5.10), and (5.13). Since temporal uncertain knowledge in (5.10) and (5.13) is automatically handled by propagation and semantic normalization, only the case where d_2 contains a value restriction as in (5.8) has to be considered. There is one rule for the operator **until** and one for **since**.

Rule 5 (Backward Propagation with Until) A world description $\mathcal{W}$ with terminology $\mathcal{T}$ and a fact 'o_1 :: d_1 **until** d_2 **at** 0' is given.[6] If

$\mathrm{a}([\mathsf{t}_1,\mathsf{t}_2]) \subseteq$ **time_compare**(**all**(r,c),d_2),
(o_1,o_2) :: r **at_all** T_2,
$\mathrm{a}(\mathsf{T}_3) \subseteq$ **time_disjoint**(c,$\mathrm{MSG}_{\mathsf{o2}}$) and
$[\mathsf{t}_1+1,\mathsf{t}_2+1] \cap \mathsf{T}_2 \cap \mathsf{T}_3 \neq \emptyset$,

is valid, the period of validity elongates in the future:

o_1 :: d_1 **until** d_2 **at_all** $[0,\max(\mathsf{T}_2 \cap \mathsf{T}_3)\text{-}\mathsf{t}_1]$ □

Rule 6 (Backward Propagation with Since) A world description $\mathcal{W}$ with terminology $\mathcal{T}$ and a fact 'o_1 :: d_1 **since** d_2 **at** 0' is given. If

$\mathrm{a}([\mathsf{t}_1,\mathsf{t}_2]) \subseteq$ **time_compare**(**all**(r,c),d_2),
(o_1,o_2) :: r **at_all** T_2,
$\mathrm{a}(\mathsf{T}_3) \subseteq$ **time_disjoint**(c,$\mathrm{MSG}_{\mathsf{o2}}$) and
$[\mathsf{t}_1\text{-}1,\mathsf{t}_2\text{-}1] \cap \mathsf{T}_2 \cap \mathsf{T}_3 \neq \emptyset$,

is valid, the period of validity elongates in the past:

o_1 :: d_1 **since** d_2 **at_all** $[\min(\mathsf{T}_2 \cap \mathsf{T}_3)\text{-}\mathsf{t}_2,0]$ □

Using these rules leads to a new problem. It is the problem of non-termination. For some constellations of facts, it is possible that these rules can be applied alternately again and again, so that the assert-time-inferences will never end. The following shows an example.

$$\mathsf{o}_1 :: \mathbf{alltime}([0,1],\mathbf{not}(\mathsf{c}_1))\ \mathbf{until}\ \mathbf{all}(\mathsf{r}_1,\mathsf{c}_2)\ \mathbf{at}\ 0 \tag{5.14}$$

$$(\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}_1\ \mathbf{at\_all\_times} \tag{5.15}$$

$$\mathsf{o}_2 :: \mathbf{alltime}([0,1],\mathbf{not}(\mathsf{c}_2))\ \mathbf{until}\ \mathbf{all}(\mathsf{r}_2,\mathsf{c}_1)\ \mathbf{at}\ 0 \tag{5.16}$$

$$(\mathsf{o}_2,\mathsf{o}_1) :: \mathsf{r}_2\ \mathbf{at\_all\_times} \tag{5.17}$$

[6] Because all **until**-terms are transformed to **until**-terms at time point 0 by the normalization, the rule is only formulated for this case.

By respecting (5.15) and (5.16), rule 5 can be applied to (5.14) . Because 'o_1 :: **all**(r_1,c_2)' can't be valid at the interval [0,1] we can deduce:

$$\mathsf{o}_1 :: \mathbf{alltime}([0,1], \mathbf{not}(\mathsf{c}_1)) \ \mathbf{until}\ \mathbf{all}(\mathsf{r}_1, \mathsf{c}_2)\ \mathbf{at\_all}\ [0,1] \quad \text{that is}$$
$$\mathsf{o}_1 :: \mathbf{alltime}([0,2], \mathbf{not}(\mathsf{c}_1)) \ \mathbf{until}\ \mathbf{alltime}([1,1], \mathbf{all}(\mathsf{r}_1, \mathsf{c}_2))\ \mathbf{at}\ 0 \tag{5.18}$$

Rule 5 can also be applied to (5.16) considering (5.17) and (5.18), and it can be entailed:

$$\mathsf{o}_2 :: \mathbf{alltime}([0,1], \mathbf{not}(\mathsf{c}_2)) \ \mathbf{until}\ \mathbf{all}(\mathsf{r}_2, \mathsf{c}_1)\ \mathbf{at\_all}\ [0,2] \quad \text{that is}$$
$$\mathsf{o}_2 :: \mathbf{alltime}([0,3], \mathbf{not}(\mathsf{c}_2)) \ \mathbf{until}\ \mathbf{alltime}([2,2], \mathbf{all}(\mathsf{r}_2, \mathsf{c}_1))\ \mathbf{at}\ 0 \tag{5.19}$$

Now rule 5 can be applied to o_1 (5.18) and thereafter to o_2 (5.19). So the interval of the **until**-terms elongates more and more. Since the validity of the role relations is not limited in the future, the alternately execution of the rules would never terminate. The world description resulting from this infinite procedure would look as follows:

$$\mathsf{o}_1 :: \mathbf{alltime}([0,+\infty], \mathbf{not}(\mathsf{c}_1)) \ \mathbf{until}\ \mathbf{alltime}([+\infty,+\infty], \mathbf{all}(\mathsf{r}_1, \mathsf{c}_2))\ \mathbf{at}\ 0 \tag{5.20}$$
$$(\mathsf{o}_1, \mathsf{o}_2) :: \mathsf{r}_1\ \mathbf{at\_all\_times} \tag{5.21}$$
$$\mathsf{o}_2 :: \mathbf{alltime}([0,+\infty], \mathbf{not}(\mathsf{c}_2)) \ \mathbf{until}\ \mathbf{alltime}([+\infty,+\infty], \mathbf{all}(\mathsf{r}_2, \mathsf{c}_1))\ \mathbf{at}\ 0 \tag{5.22}$$
$$(\mathsf{o}_2, \mathsf{o}_1) :: \mathsf{r}_2\ \mathbf{at\_all\_times} \tag{5.23}$$

But now it is obvious that it is an inconsistent world description. Because of (5.20) and the semantic of **until**, some time after the time point 0 '**alltime**(+∞,**all**(r_1,c_2))' is valid. But this is in opposition to (5.21) and (5.22).

This was only a simple example of a cyclic rule usage. With rule 6, such cycles can be constructed too. It is yet to clarify, if such non-terminating usage of the rules can only exist in an inconsistent world descriptions.

It is the same problem as in the TBox. It is not sure that the usage of some rules of the semantic normalization will terminate.[7] To handle such situations, an upper limit for the number of rule applications was set. If this limit is exceeded, the semantic normalization stops and a warning is displayed for the user.

The solution for the ABox is analogous. After every user-tell, the consequences for all objects are calculated. It is counted how often the rules 5 and 6 are used. After exceeding an upper limit, these rules will not be used to produce new consequences. Therefore it is sure that this procedure will terminate in every case.

Of course, these solutions are only provisional. We have yet to clarify, how such cyclic rule applications can be discovered quickly and safely, and how they have to be handled. Especially it has to be examined, if they only coexist with incoherence or inconsistency.

[7] see [Fischer 92a, p. 20]

5.2 ABox Asks

Since the most inferences are calculated at assert-time, in this section only abstraction is described. It is the only inference calculated at query-time. Because of the pre-computations, the calculations of the answers are not so hard to realize. So they are not described here but in the next chapter.

5.2.1 Abstraction

The abstraction of closed role-filler-sets belongs to the backward propagations which exists also in non-temporal systems. Look at an example:

(Mary,Robert) :: has_child (5.24)

(Mary,Ann) :: has_child (5.25)

Mary :: **atmost**(2,has_child) (5.26)

Robert :: study (5.27)

Ann :: study (5.28)

The formulas (5.24) to (5.26) imply, that Mary has a closed role-filler-set concerning the role 'has_child', because all role-fillers are known. Together with (5.27) and (5.28) we can infer 'Mary **:: all**(has_child,study)'. If there is a closed role-filler-set of an object o, a value 'restriction' for o can be constructed by a generalisation of the MSGs of the role-fillers. A generalisation of two concepts is the most special concept subsuming these two. So the calculated value restriction is the most special concept which is valid for *all* role-fillers.

Indeed, in normal μBACK there is a high expense for calculating the abstraction, and a small profit of information. In temporal μBACK we need a higher expense because there can be many closed role-filler-sets per object and role for example, and not only one. But without regard to this, it is unknown how to calculate the generalisation of two temporal concepts.

For those reasons, the abstraction is not calculated at assert-time but at query-time. So no generalisations have to be calculated. But if the query contains a value restriction, the answer can't be only intensionally computed by a simple call of **time_compare** as written in [Fischer 92a, Fischer 92b], but it must also be extensionally calculated.

For example the answer of the query '**when_is** o **::** **all**(r,c)' can't be simply calculated by a call of the function **time_compare**(**all**(r,c),MSG_o). First, we have to examine if there is a closed role-filler-set, and when all role-fillers are instances of c. For this period, a value restriction can be added to the MSG, and then the answer can be computed with a call of the function **time_compare**.

At last, I will give a short example for the temporal abstraction in the context of our little world description.

(Mary,John) :: spouse **at** 1991 (A9)

John :: dead **at** 1992 (A10)

Mary :: unmarried **at** 1992 (A11)

The answer of '**what_is** Mary **at** 1992' contains the concept 'widowed' too. Because of (A7) and (A9), a closed role-filler-set exist in 1991. At this time point, we can abstract 'Mary **::** **all**(spouse,**nextt**(dead))'. Since she is married in 1991 and unmarried in 1992, she is widowed in 1992.

Chapter 6

Implementation

In this chapter I will give an overview how the algorithms are implemented in the system. After giving a general overview, I describe the realization of the assert-time-inferences and the query-time-inferences.

Figure 6.1 shows the structure of the temporal ABox and its connections to the whole system. The modules `h_q` and `h_c` contain specific information for Quintus-Prolog and CProlog, so that the system can run with both versions. Since the ABox calculates with timesets, the corresponding module is used. As mentioned above, the ABox needs functions of the TBox. The TBox itself is structured in several modules but this is not in the focus of our interest. Its architecture is described in detail in [Fischer 92b, p. 30]. Because of the required input /

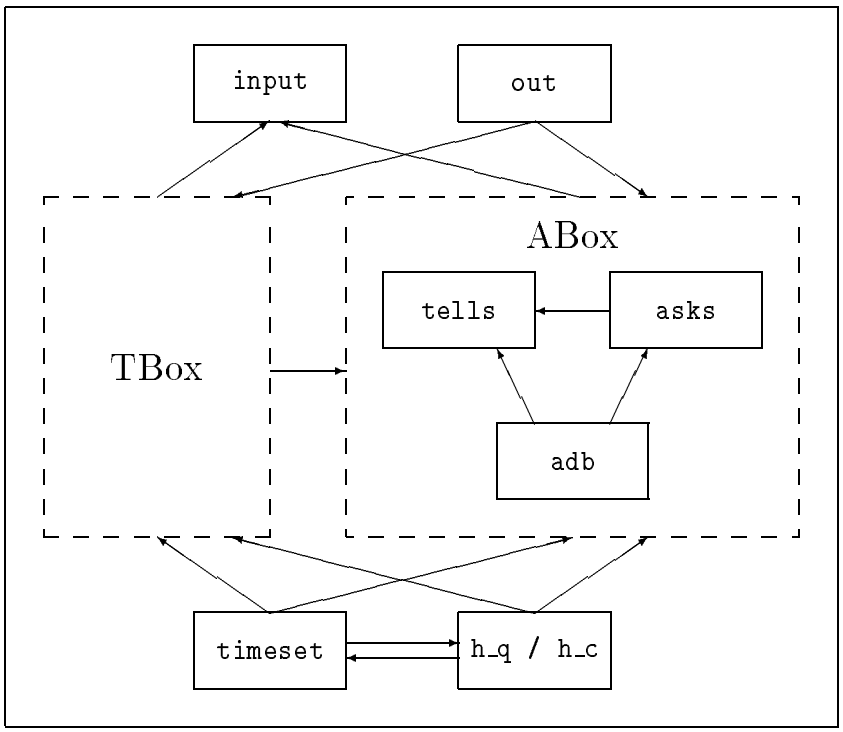


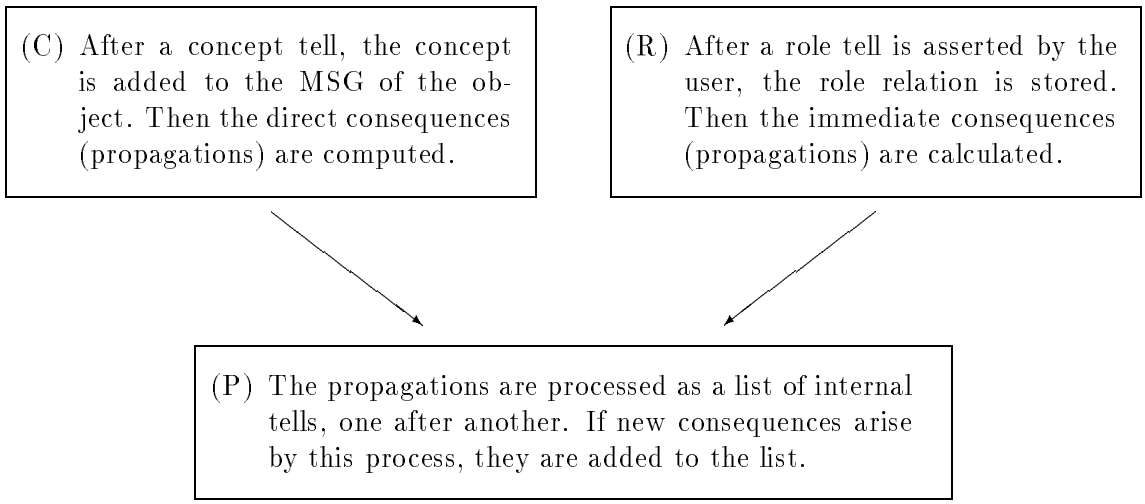
Figure 6.1: Structure of the Temporal ABox

output, the ABox is also connected with these modules.

The ABox itself consists of three modules. The database operations in `adb` are used as bases for the other two modules. Here it is stated how the dates are stored and how they can be accessed. The largest module is `tells`, where all ABox tells are processed and all assert-time-inferences are calculated by the use of the rules 1 to 6. In `asks`, there are the algorithms to answer the queries. The actions done by the two modules `tells` and `asks` are now described in detail.

6.1 ABox Tells

The general process after an ABox tell looks as follows:



This structure could be taken from the normal μBACK (see [Deumer, Neuwirth 91]). In the next subsections, the steps (C), (R) and (P) are explained.

6.1.1 Concept Tells

After the user asserts a concept tell of the form $\mathtt{o} :: \mathtt{c}\langle at \rangle \mathsf{T}$, the following steps are executed:

(C1) The concept term is transformed by the normalization into the internal TBox representation. If the term is syntactical incorrect, the normalization stops and an error message occurs.

(C2) The expression '$\ldots\langle at \rangle \mathsf{T}$' is transformed into a timeset.

(C3) The input of the user is stored for later use.

(C4) An internal tell (see (CI)) is executed, which adds the new information to the MSG of o and calculates the immediate consequences for other objects.

I have named it 'internal tell' (CI), because from here on the internal calculated tells (the propagations) and the user-asserted tells are handled in the same way. The arguments of (CI) are a name of an object, a concept normal form and a timeset which decides when the object is an instance of the normal form. The result is a list of new consequences calculated by the

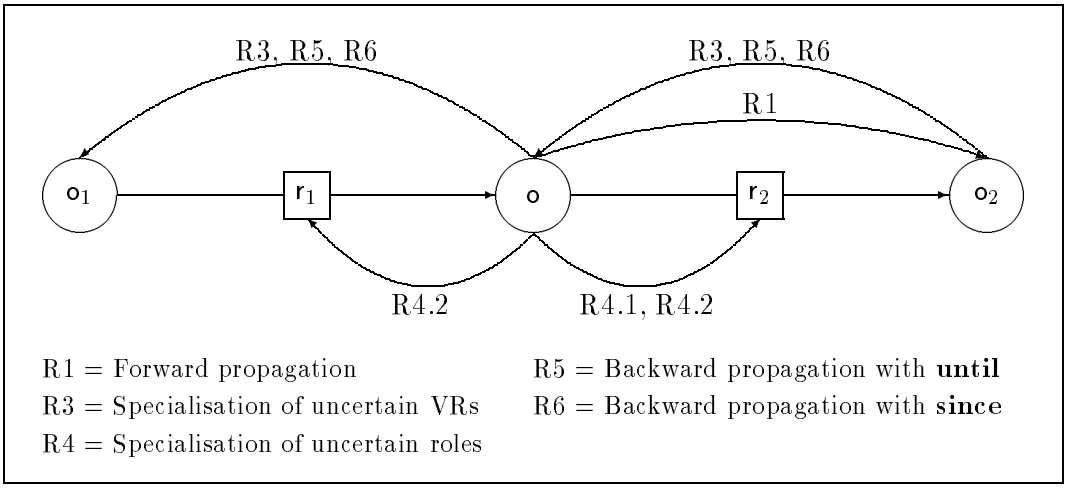


Figure 6.2: Possible Rule Executions after a Concept Tell for the Object o

rules 1 to 6. In figure 6.2 the possible rule executions after a Concept Tell for the Object o are shown in a graphical way.

All internal tells arise from consequences of a user-tell. The user-asserted tell and its consequences are stored temporarily. Only if all consequences are processed successfully and the system did not become inconsistent, the calculated results are stored permanently in the knowledge base.

(CI1) The tell is examined whether it contains new information for the object or not. If there is no new information, (CI) finishes. The list with the new calculated consequences is empty.

(CI2) The new information is added to the MSG of o.

(CI3) It is tested, if the new information is influenced by the role-fillers using the rules 3, 5 or 6 (arrow from o_2 to o). If it is influenced, the MSG is changed accordingly. It is also tested, if an uncertain period of the belonging roles can be restricted by using rule 4.1 or 4.2 (arrow from o to o_2).

(CI4) If the MSG of o became incoherent, all consequences of the inducing user-tell are cancelled, and the procedure is finished with an error message.

(CI5) The new MSG is stored temporarily, that means the old information is preserved, so that the changes can be cancelled.

(CI6) The forward propagations are calculated with rule 1 (arrow from o to o_2).

(CI7) The backward propagations to the inverse role-fillers are calculated using rules 3, 5 and 6 (arrow from o to o_1). Moreover it is examined, if rule 4.2 can be used for roles, where the object o is a role-filler (arrow from o to r_1).

(CI8) The new computed consequences are returned by a list of internal tells.

Normally at forward propagation only the new information for this tell is considered, because the existing information has been considered at a prior tell. But if the semantic normalization

is switched on in the TBox, older information is influenced by the new one. For that reason at forward propagation, always the whole MSG has to be considered.

6.1.2 Role Tells

In the following, the steps after a role tell $(\mathsf{o}_1,\mathsf{o}_2) :: \mathsf{r}\langle at\rangle\mathsf{T}$ are described.

(R1) The expression '$\ldots\langle at\rangle\mathsf{T}$' is transformed into a timeset.

(R2) The input from the user is stored for later need.

(R3) An internal role tell is called which stores the new information and calculates the direct consequences for other objects, see (RI).

For role tells, there is a so called 'internal tell' too, which returns a list with new arising consequences.

(RI1) It is examined, if the new tell contains a new information. If not, (RI) finishes. The list with new arising consequences is empty.

(RI2) The given period of the new role relation is united with a possible existing period, and then stored temporarily.

(RI3) The forward propagations which are now possible from o_1 to o_2 are calculated (rule 1).

(RI4) The new backward propagations from o_2 to o_1 and sometime-restrictions are computed with rules 3, 4.2, 5 and 6.

(RI5) The role-fillers of the new period are calculated with rule 2, and returned as Atleast-restrictions.

(RI6) Possible restrictions of uncertain role relations are computed using rule 4.1.

(RI7) All new calculated consequences are returned as a list of new internal tells.

6.1.3 Propagations

For the minimization of run-time, the number of internal tells should be as small as possible. Especially if the semantic normalization is switched on, many calculations are repeated again and again. Therefore the list of the propagations which have to be processed is optimized by combining tells for same objects. This procedure was also taken from normal μBACK.

A concept propagation consists of four elements, the object, the concept, the period of validity, and an indication, showing whether the propagation was created by rule 5 or 6. After every user tell, it is counted for every object how many propagations are created by these rules. As mentioned in section 5.1.4, such propagations are not further considered after exceeding an upper limit because of the problem of non-termination.

A role propagation consists of four elements too: the two objects, the role, and the period of validity. Since no role propagation can be created by rule 5 or 6, no indication is necessary.

(P1) If the next propagation is a concept propagation created by rule 5 or 6, and the upper limit of such propagations is already exceeded, this propagation is not considered. Continue with (P5).

(P2) The next internal role or concept tell is executed. Perhaps new consequences arise. If there is a propagation produced by rule 5 or 6, the corresponding counter is incremented.

(P3) If the new propagation leads to an inconsistency, (P) fails. Then all temporarily stored data are deleted and the user-asserted fact is rejected showing an error message. Now the system is in the same state as before the user tell.

(P4) New propagations from (P2) are added to the actual list.

(P5) (P) is called recursively.

(P6) If the list is empty, that happens when all necessary propagations are successfully processed, the temporarily stored data are transferred in the real knowledge base. The 'rule counters' are cleared to 0.

Procedure (P) will always terminate, because cyclic term definitions are not allowed, already known information will not produce new propagations (see (CI1) and (RI1)) and the usage of the rules 5 and 6 is restricted by an upper limit.

6.2 ABox Asks

In [Fischer 92b, p. 32] M. Fischer has described simple algorithms for the operators **when_is**, **who_is** and **what_is**. They are based on simple TBox queries with **time_compare** and **subsumes**. But this simple procedure could not be realized, because the precondition that all needed information is stored in the MSG of the objects, is not satisfied, since the abstraction is only done at query-time. For that reason, this procedure had to be modified.

The two functions **isinstance**(O,Cnf,TS) and TS = **abstract_instance**(O,Cnf) are the heart of the algorithms for answering concept queries. Considering a possible abstraction, they calculate if or when an object O is instance of a concept normal form Cnf.

The functions are divided into two parts. In the first part, the abstraction of O concerning the value restrictions of Cnf is computed. The results are added immediately to the MSG of O. In the second part, it can be examined with a simple call of **time_compare**, if or when the instantiation is valid. The abstraction contains the following steps:

(A1) If there is no value restriction in Cnf, no abstraction is possible and (A) finishes.

(A2) If there are primitive or negated primitive concepts in Cnf, which do not exist in the MSG, an abstraction is not necessary, because O can't be an instance of Cnf at any time point. The additional expense for calculating the primitive concepts contained in Cnf and MSG is very small, since they have to be examined anyhow for existing Atmost- and value restrictions.

(A3) For every value restriction, all closed role-filler-sets are calculated.

(A4) For the period, where all objects of a closed role-filler-set are instance of a corresponding value restriction, an **all**(r,VR) can be added to the MSG.

The last step, examining when the role-fillers are instances of VR, is a recursive call, because for this examination possible abstractions have also to be considered. Since **all**-terms are not endless deep nested, the procedure will terminate in every case.

Now the concept queries can simply be answered with the help of these functions:

- o ? c⟨*at*⟩T:
  - The expression '...⟨*at*⟩T' is transformed into the timeset TS and c is normalized to Cnf.
  - The result is calculated by **isinstance**(o,Cnf,TS).
- **when_is** o :: c:
  - The concept c is normalized to Cnf.
  - The output of **abstract_instance**(o,Cnf) is returned.
- **who_is** c ⟨*at*⟩ T:
  - The concept c is normalized to Cnf and '...⟨*at*⟩T' is changed to the timeset TS.
  - All objects are examined by **isinstance**, if they are instances of Cnf at TS.
- **what_is** o ⟨*at*⟩ T:
  - The user has signed a subset of the user-defined concepts as the set which has to be examined. Momentary this is done by a filter predicate which exclude the p_-concepts from all user-defined concepts (see page 17).
  - It is tested which of these concepts are instantiated by o in the given period.
- **concepts_of** o:
  - For all concepts which have to be examined (see above), it is computed with **abstract_instance**, when o is an instance of them.
  - These periods are broken into intervals and all concepts for the same interval are united.
  - For the output, this list of intervals and concept-sets is sorted chronologically.
- **instances_of** c:
  - The concept c is normalized to Cnf.
  - For all objects it is calculated with **abstract_instance** when they are instances of Cnf.
  - These periods are broken into intervals and all objects for the same interval are united.
  - For the output, this list of intervals and object-sets is sorted chronologically.

Now we come to the implementation of the enhanced query-scheme 'O **::** C **at** TS'. First, I try to transform all of the three expressions into a unique form:

o	→	O **in** [o]
O	→	O **in** ⟨*list-of-all-objects*⟩
O **in** OList	→	O **in** OList
c	→	C **in** [c]
C	→	C **in** ⟨*list-of-all-TBox-concepts*⟩
C **in** c	→	C **in** ⟨*list-of-all-TBox-concepts-subsumed-by-*c⟩
C **in** CList	→	C **in** CList
TS	→	TS
T	→	T **in** [-∞,+∞]
T **in** T_1	→	T **in** T_1

In all but one cases, the expressions can be transformed into those with the operator '**in**'. For all objects of the object list, that period is calculated with the function **abstract_instance**, where it is instance of an concept of the concept list. This period is intersected with the given certain pointset. If the intersection is not empty, it is a valid substitution.

But there is one exception, if a certain timeset is given. In this case, for all objects of the object list it is tested with **isinstance**, whether it is instance of a concept of the concept list at the given period or not.

The implementation of the role-queries are not described here. They can be realized by simple comparison of timesets. The enhanced scheme for roles is implemented analogously to the scheme for concepts.

Chapter 7

Evaluation and Outlook

7.1 Incompleteness

As mentioned above, not only the user-asserted explicit knowledge is considered for ABox queries, but also the concealed implicit knowledge. This knowledge is logically implied by the given facts. The logical implications have to be expressed by inference rules which should be correct and complete, that means all inferences should be logically implied and all logical implications should be calculated by the inference rules. Which sorts of logical implications are possible depends on the used language, and so the necessary inference rules are determined.

In the ABox, the rules 1 to 6 and the abstraction rule are correct, but some of them are incomplete. Therefore, some implicit information is not computed and can't be considered for answering queries. The rules 3, 4.2 5 and 6 are incomplete (see section 5.1.3 and 5.1.4). The incompleteness arises at that point, where the validity of an uncertain **all**(r,c)-terms has to be restricted. The restriction is done for those periods, where c is in contradiction to the description of a role-filler. As described in section 5.1.3, the contradiction doesn't need to arise at this point. Any contradiction between c and the world description is sufficient. This universal possibility is not considered, because I don't know any efficient proceeding to calculate *when* a fact is in contradiction to the knowledge base. A complete but inefficient proceeding is described in section 5.1.3.

Additional incompleteness arise because of the limitation of the number of rule executions for R5 and R6.[1] This upper boundary was introduced to avoid the non-termination of cyclic rule executions. But the consequence is that in some cases the validity of some **since**- and **until**-terms are not completely calculated.

Moreover there is the incompleteness of the TBox. In [Fischer 92a, p. 27] is described, in which cases the function **time_compare** is incomplete. Because this function is used for calculating propagations and for answering ABox-queries, the results of the ABox are influenced by this incompleteness.

7.2 Performance

When we look at the performance, we can distinguish two domains: the behaviour at assert-time when the world description is entered and the behaviour at query-time.

[1] see section 5.1.4 and 6.1.3

The behaviour at assert-time is determined by the number of necessary assert-time-inferences. It can be influenced in a positive manner by considering the following relationships:

- In a world description without any role relation no ABox inferences are necessary.
- If no value restrictions are used, only the rules 2 and 4.1 can be executed.
- The avoidance of uncertain knowledge reduces the executable rules to the rules 1, 2 and the abstraction.

A consideration of these dependencies leads to a better performance and reduces incompleteness.

The time which is needed to calculate the answers for ABox-queries depends on several factors. For simple queries such as '`o ? c` **at_all** ⊤' or '**when_is** `o :: c`' the main factor is, whether an abstraction has to be computed, that is whether `c` contains value restrictions with one or more belonging closed role-filler-sets, or not. In this case, several normal forms have to be compared and if necessary additional abstractions have to be done. Since more complex queries concerning some objects or concepts are reduced to several simple queries, this factor is here more important.

Because in the TBox, and therefore in the ABox too, no classification is done, and indexing as in the normal μBACK could not be simply transferred to temporal system, simple algorithms for the queries **who_is** and **what_is** are implemented, where all objects of the world description or all concepts of the TBox are examined. Therefore the performance is highly influenced by the number of objects and defined concepts.

7.3 Advantages of Restricted Languages

Because of the existing incompleteness and the performance we have to think about more restricted languages with corresponding algorithms.

The temporal term language was built by adding temporal operators to the language $\mathcal{NTF}$. If only the representation of *certain* temporal knowledge is allowed, that is only the operators **alltime** and **at_all** are added, we get a very simple temporal system with a temporal indexed ABox. As in normal μBACK, only forward propagation (R1), Atleast-abstraction (R2), and abstraction of closed role-filler-sets has to be computed. Moreover, the TBox function **time_compare** can be implemented easier and more complete.[2] So we get a nearly complete system with a better performance.

Based on this system for processing temporal certain knowledge, the query language could be enhanced with additional temporal operators without modifying the algorithms of the TBox and for the ABox inferences. Only the interpreter for the queries has to be modified.

If we want to represent uncertain temporal knowledge with the operators **sometime** and **at_some**, the situation gets worse. New incompleteness arises in TBox and ABox and the performance degrades by additional inference rules. Because the handling of nested **sometime**-terms is difficult for the TBox, it is better to omit this operator. But then nothing changed in the ABox, the rules 3 and 4 are needed furthermore.

[2] see [Fischer 92a, p. 29]

7.4 Conclusion

Together with the works of M. Fischer, it was shown that it is possible to construct and to implement a temporal terminological representation system according to the disposition of K. Schild. In the TBox, a temporal terminology can be built and used for the temporal object descriptions in the ABox. By the implemented inference rules, the system is able to deduct 'new' implicit knowledge. A temporal query language allows to ask simple and combined ABox queries. So it is possible to ask very directly for the desired information.

But it was also shown that there are some problems to handle the full language. Restrictions arise by the incompleteness and the performance. Moreover it is necessary that the user has a well founded understanding of the language to get the desired results. If possible, a restricted language should be used, for example, a language only for temporal certain knowledge.

Beside of algorithms for avoiding incompleteness and for increasing efficiency, there are other interesting possibilities of enhancement, for example the integration of role operators, a temporal D(efault)Box to handle persistence or a temporal I(mplication)Box to process temporal rule knowledge.

With this implemented temporal representation system, additional experiences can be gathered concerning the practicability of this disposition, and it can serve as base for future enhancements.

Bibliography

[Allen 83] James F. Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM* 26(11), S. 832–843, 1983

[Brachman, Schmolze 85] Ronald J. Brachman, James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science* **9** (2), S. 171-216, 1985

[Deumer, Neuwirth 91] Wolfgang Deumer, Andreas Neuwirth. Implementierung einer ABox fr μBACK. Semesterarbeit im Seminar 'Terminologische Wissensrepräsentation' am Fachbereich Informatik, Technische Universität Berlin, 1991

[Fischer 91] Martin Fischer. Integration von temporalen Operatoren in ein terminologisches Repräsentationssystem. Studienarbeit am Fachbereich Informatik, Technische Universität Berlin, 1991

[Fischer 92a] Martin Fischer. Inferenzalgorithmen für ein temporales terminologisches Repräsentationssystem. Diplomarbeit am Fachbereich Informatik, Technische Universität Berlin, 1992

[Fischer 92b] Martin Fischer. The Integration of Temporal Operators int a Terminological Representation System. *KIT Report 99*, Berlin: Fachbereich Informatik, Technische Universität Berlin, 1990

[Nebel 90] Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems.* LNAI 422, Springer, Berlin Heidelberg, 1990

[Neuwirth 92] Andreas Neuwirth. Inferenzen für temporale Objektbeschreibungen in einem terminologischen Repräsentationssystem: Entwurf und Implementation. Diplomarbeit am Fachbereich Informatik, Technische Universität Berlin, 1992

[Peltason et al. 89] Christof Peltason, Albrecht Schmiedel, Carsten Kindermann, Joachim Quantz. The BACK System Revisited. *KIT Report 75*, Berlin: Fachbereich Informatik, Technische Universität Berlin, 1989

[Quantz, Kindermann 90] Joachim Quantz, Carsten Kindermann. Implementation of the BACK System Version 4. *KIT Report 78*, Berlin: Fachbereich Informatik, Technische Universität Berlin, 1990

[Schild 91a] Klaus Schild. A Correspondence Theory for Terminological Logics. *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, S. 466–471, Sydney, Australia, 1991

[Schild 91b] Klaus Schild. A Tense-Logical Extension of Terminological Logics. *KIT Report 92*, Berlin: Fachbereich Informatik, Technische Universität Berlin, 1991

[Schmiedel 88] Albrecht Schmiedel. A Temporal Constraint Handler for the BACK System. *KIT Report 70*, Berlin Fachbereich Informatik, Technische Universität Berlin, 1988

[Schmiedel 90] Albrecht Schmiedel. A Temporal Terminological Logic. *Proceedings of the 9th Conference of American Association for Artificial Intelligence*, S. 640–645, Boston, Mass., 1990

[Shoham 87] Yoav Shoham. Temporal Logics in AI: Semantical and Ontological Considerations. In *Artificial Intelligence, volume 33*, S. 89–104, 1987