Modelling the Progressive and Coercion using the Event Calculus

Bachelor Thesis

Maria Staudte

Institute of Cognitive Science
University of Osnabrück

First examiner: Prof. Dr. Kai-Uwe Kühnberger
Second examiner: Dr. habil. Helmar Gust

Osnabrück, September 2004

Abstract

This bachelor thesis summarises Hamm and van Lambalgen's book "The proper treatment of events". The main points are addressed to provide an overview of the Event Calculus and its usage in natural language semantics. The Event Calculus is a formal system derived from the Situation Calculus that allows for the representation of overlapping events and partial changing objects. Its ontology comprises three types: events, fluents and time points. Thus, natural language expressions are decomposed into such elements construing a scenario to represent what is claimed to be the sense of that expression. This thesis exemplifies several principles and mechanisms involved in this representation using mainly verb phrases. Further, a process called 'coercion' is described which labels the phenomenon that one verb can occur with different aspectual classes, here distinguished according to Vendler. Coercion is represented by the Event Calculus by modifying a 'default' scenario of that expression and thereby creating a related scenario, thus a relation between the two aspectual classes is established. The progressive is a tense that often induces change of aspectual class and therefore is an excellent sample for investigations on coercion. Because this algorithmic approach is claimed to model sense and reference and thus to replace the conventional possible worlds semantics, a discussion on this issue follows the detailed analysis of the single aspects.

Contents

1 Introduction

Natural language semantics is traditionally based on the definition of meaning via possible worlds, i.e. the proposition of an expression is modelled by the set of worlds in which the expression is true (see 7.1). For many linguists, and so for Hamm and van Lambalgen, this approach does not provide a suitable model with cognitively adequate explanations. That is why they suggest to return to the origins of the distinction between sense and reference formed by Frege and Carnap and focus on the computational aspect of *sense* motivated by the mathematician Yiannis Moschovakis. By means of the Event Calculus they model natural language expressions and use Constraint Logic Programming to implement this formalisation. The resulting algorithms are considered to represent the senses of the respective expressions while the result of their computation construes the references of the expressions. The aim of this bachelor thesis is to present Hamm and van Lambalgen's approach and to discuss the idea that an algorithm shall be the sense of an expression. To support this discussion, some programs are presented in order to illustrate and evaluate their theory on examples.

In detail, this thesis is organised as follows: In the next subsection, Moschovakis' idea is outlined followed by an embedding of the respective theory in Cognitive Science in section 2. Moreover, the motivations and origins of the framework that Hamm and van Lambalgen construct are briefly explained and I outline the points that justify the interest for Cognitive Science. In section 3, I introduce the formal system that is expected to constitute an appropriate model for the meaning of expressions and their references, the Event Calculus. The consecutive section provides a short description of the programming environment. After the formal and abstract presentation section 5 follows which is devoted to explaining and illustrating how the Event Calculus is applied. It includes examples of verb phrases and an elaborate description of their internal structure. This internal structure motivated the distinction between the particular aspectual classes and these are addressed individually and in detail to pave the way for the subsequent section. In the latter a transformation mechanism is presented which is known as 'coercion' and which names the assumption that a verb is forced from one aspectual class into the other by its context. The modelling of this mechanism is illustrated in section 6 mainly using the progressive inducing the change. In section 7 I juxtapose Frege's 'Sinn' with the concept of 'intension' which was formed by Carnap. This is to provide the reader with some ground knowledge for the adjacent discussion of the claim that an algorithm, in this case a prolog program realising an Event Calculus scenario, is the sense of the expression from which the scenario is derived. Finally, section 8 discusses the presented material.

$$\begin{tikzcd}
\text{Term (A)} \arrow[r] & \text{Sense (A)} \arrow[r] & \text{Denotation (A)}
\end{tikzcd}$$

Figure 1: *Frege's idea*

$$\begin{tikzcd}
 & \text{A} \arrow[dr] & & \\
\text{Programs:} & \text{B} \arrow[r] & \text{Algorithm (A)} \arrow[r] & \text{Denotation (A)} \\
 & \text{C} \arrow[ur] & &
\end{tikzcd}$$

Figure 2: *Moschovakis' analogue*

Moschovakis' Motivation

Hamm and van Lambalgen's work has been motivated by Yiannis Moschovakis who is a researcher in the field of mathematical logic. His original intentions have been to find an appropriate definition for the term algorithm and to relate the concept 'algorithm' to the concept of synonymy. The criteria for deciding whether synonymy between two terms exists go back to the distinction of sense and reference of these terms. Thus, Moschovakis' approach is relevant for philosophy and linguistics as well where a dominating model for sense and reference is based on possible worlds (see section 7.1 for a detailed description). He defines synonymy to be a 'faithful translation' between terms, this corresponds to the concept of identity disregarding structural differences. In natural language, such structural differences can occur as different expressions which have the same sense or intension and where the extension or reference is identical. Moschovakis adapts Frege's distinction between sense and reference and maps the former to the concept of an abstract algorithm which computes the latter. In other words, different programs implement the same algorithm like different terms may have the same sense and this algorithm determines the denotation of the program corresponding to the sense that determines the reference of the terms.

This abstract algorithm can be implemented by several programs which correspond to the different terms which share one sense (see figure 2). Formally this relation can be expressed as follows:

> Assume that a first-order (sorted) structure **A** for the *Formal Language of Recursion* (FLR) is given. A denotation $den_{\mathbf{A}}(\bar{x}, \phi)$ is defined recursively with respect to **A**, where ϕ be an FLR-formula all of whose free variables are among $\bar{x}$. An intension $int_{\mathbf{A}}(\bar{x}, \phi)$ is then defined to be the algorithm which computes $den_{\mathbf{A}}(\bar{x}, \phi)$. [1]

[1] Adapted from (Hamm & van Lambalgen, 2003b) and reformulated. For further tech-

The term $den_{\mathbf{A}}(\bar{x}, \phi)$ specifies the denotation of a formula ϕ which includes variables from $\bar{x}$. This denotation of ϕ is, of course, dependent on the frame work and as such the structure **A** specifies the environment in which ϕ exists and is evaluated. The intension of ϕ is given by $int_{\mathbf{A}}(\bar{x}, \phi)$ which is defined to be the algorithm that computes the denotation of ϕ.

2 Meaning and Tense

2.1 Cognitive Time

Hamm and van Lambalgen commence their work by pointing out several aspects of natural language and its relations to other abstract cognitive processes like planning. Here we find the important embedding in Cognitive Science. However, apart from the broader Cognitive Science research interest the link to planning is worth to be examined in depth since much research has been done in this field which can and will contribute a great deal to the investigations of (the semantics) of natural language. In the following, I will briefly summarise the main arguments that support the planning aspect of language. This is to provide the appropriate embedding and justification for the theory to come.

The question is why humans and how humans experience time because if it can be investigated what the human mental representation of time looks like and what its functionality is, then this will roughly apply to the linguistic representation as well.

Psychologists typically distinguish three different cases when investigating time:

1. time as duration,
2. temporal perspective,
3. time as succession.

Time as duration is at least in English or German not grammaticalised, rather is the duration of e.g. five minutes represented by a corresponding lexicalisation (*"for* five minutes"). *Temporal perspective*, however, is much more interesting since it involves past and future and ordering of events. Perspective exists only if someone perceives events and time and thus, takes on a subjective point of view. From there, from the deictic *now*, the subject perceives the past as events that have happened and that it can retrieve from its memory and it can anticipate events that may occur in the subjective future.[2]

nical details and an elaborate explication of the mathematical basis for this statement see (Moschovakis, 2003).

[2]The deictic *now* is neurologically shown to be an instant of around 3 seconds. This contributes to the hypothesis that the *present* is durative and merges with the past at one end and with the future at the other. (Hamm & van Lambalgen, 2004)

The strongest argument that Hamm and van Lambalgen drive forward here is the experiment by Trabasso and Stein mentioned in (Russel & Norvig, 2003) which supports and perspicuously illustrates the congruence of language and planning. In this experiment, children of different ages as well as adults narrate a story from presented pictures. It shows that the children only learn to narrate using future and past when simultaneously developing an understanding for the causality of the depicted events. But not only the causal relation plays a role, moreover, the children had to learn to recognise aims and intentions of the characters in the story first in order to successfully use tenses. This evidently indicates that the representation and the use of tenses is dependent on the understanding of causality as well as the capability to mind-read. (Note: Mind-reading is the term used for the typically human ability to put oneself mentally into the position of someone else. Interestingly, only humans can also *imagine* themselves to be in another time than the deictic *now*.) If one now recalls that planning is the search for a sequence of actions that leads to the goal, one will agree that planning consists of (a) realising what the goal state is, i.e. what is aimed at, and (b) what causes this goal state (to then perform the appropriate action sequence). Easily, one can now merge the concept of a goal with the future tense, e.g. "what is to be achieved lies in the future", and merge the concept of a plan structure with the past, because the past provides a data base of experiences which may be relevant for what is yet to come. Consequently what children learn when they grow older is a cognitive ability called planning and it is not a mere coincidence that the usage of tenses in their mother tongue happens to develop concurrently.

Time as succession seems to be included in the concept of temporal perspective but in fact this notion is somewhat more objective and is *constructed* from the temporal perspective. This means that, once a sequence of events could be constructed, it is independent from subjective perspective and only due to truth. According to Hamm and van Lambalgen, a representation of temporal succession is created by using temporal perspective in the following way:

In the present or deictic *now* the event e happens and is stored in (working) memory. Simultaneously an associated event p is retrieved and thus recalled while event f will be anticipated. The recall- and anticipate-relation therefore determines the *precedes* relation: if p is recalled then p precedes e and if f is anticipated then e precedes f. Hence, a sequential order of events can be established (p *precedes* f). It should be emphasised that the *precedes* relation treats events as a whole and does not allow for an exact starting or end point or overlapping. To my mind, this representation of the construction of time as succession is too crude and not suitable to support the decomposition of events that the Event Calculus carries out in order to represent an event. Nevertheless, the brief elaboration above ought to have shown the close relation of planning to language and therefore why it is cognitively plausible to use an approach based on planning when dealing with natural language semantics.

2.2 Tense and Grammar

It was already mentioned that time as duration is not grammaticalised as opposed to temporal perspective. The grammaticalisation of the latter is at least in English mainly concerned with planning, i.e. with the relation between goals, plans and actions, and less with the true temporal succession of events. This becomes obvious when looking at a few examples of the English future tense below. For instance, it can be distinguished between future events (per se) that will happen regularly with or without human intervention and events that may happen and/or can be achieved by executing a sequence of actions like a goal can be achieved by executing a plan. To express the former, present tense is used as in "The sun rises tomorrow at 5 am." whereas future tense is used to express the latter: "It will rain tomorrow." or "I am going to go to Berlin tomorrow.".

The future tense can be further differentiated by the prospect of an intervening event that may prevent the execution of the plan ('to be going to' is used) and by explicitly excluding the possibility for obstacles ('will' is used). Consider the following examples:

1. It will rain tomorrow.
2. ?It is going to rain tomorrow.
3. *It rains tomorrow.

These sentences nicely illustrate the above principles. In 1 and 2 no intervention is anticipated and that is why the use of "be going to" is somewhat awkward without any further remark on why it may not rain.

1. I am going to go to Berlin tomorrow. Hopefully I won't miss the plane.
2. *I will go to Berlin tomorrow. Hopefully I won't miss the plane.
3. I will fly to Berlin tomorrow unless the weather forbids it.

It is clear in 1 and 2 that the speaker is leaving the option for an event to interrupt the plan. While 1. works fine with "be going to", "will" causes some problems. However, as can be seen in 3. "will" can be used even though the *preconditions* for the plan are yet uncertain.

Another interesting tense in English grammar is the present progressive which is a composed tense like all progressive forms in contrast to simple tenses. Here we find that events in the progressive form cannot be treated holistically. Consider for instance:

a. John crosses the street.
b. John is crossing the street.

and in the past:

a. John crossed the street.
b. John was crossing the street (when a truck hit him).

In the present both forms yield approximately the same meaning whereas the difference in the past is remarkable. The simple past tense treats the event as a whole and entails the result state that John has crossed the street. However, the past progressive differentiates between the action, the goal event and the result state such that "John has crossed the street" cannot be inferred from b. This is known as the imperfective paradox and will be investigated in detail in 6.1.

2.3 Meaning and context

Discourse involves a more global ordering of events, taking into account several sentences instead of only one. Hamm and van Lambalgen address this issue when arguing towards a semantics based on algorithms in the Event Calculus which I will introduce in detail in the next section. They criticise that conventional approaches only take the particular expression itself into account. In natural language, however, it is often the case that a subsequent sentence alters the meaning, i.e. the *sense*, of the expression in question retroactively in quite a delicate way. Consider, for instance, the utterance

1. John was crossing the street. At this moment, he saw Mary and walked faster.

2. John was crossing the street. He then got hit by a truck.

In 1., the meaning of the first sentence is that of a person called John walking across a street and probably reaching the other side. By adding the second sentence, the crossing process is slightly accelerated. In 2. adding the second sentence reduces the original meaning of *crossing the street* since the final event of reaching the other side of the street will never happen.

Furthermore, causal relations between sentences are not dependent on the order of the sentences but on their meaning as can be seen on those examples here:

1. John went up the stairs. He closed the door.
2. John closed the door. He went up the stairs.

3. John walked up the stairs. He went to bed and slept.
4. *John went to bed and slept. He walked up the stairs.

5. John entered the room. He broke the vase.

6. John broke the vase. He let it fall.

In 1. and 2. one can reverse the order of the sentences and obtain an intuitive reversion of the order of events. This may suggest that the temporal order of events is dependent on the linear order in which the sentences are. But as shown in 3. and 4. this is not always the case. The plausibility of the expressed proposition here is essential and it determines that the sequence in 4. cannot be semantically correct since a person cannot be sleeping and walking up the stairs simultaneously (disregarding sleepwalking). Sentences in 5. and 6. are even more confusing because here the order of the sentences is oppositional to the temporal order. This happens merely because a causal relation exists between the sentences but not necessarily a sequential one. So the first sentence in 6. states what happened and the second provides the cause for the event. The reverse in 5. yields again a sequential notion.

3 The Event Calculus

3.1 The Ontology

The Event Calculus developed by Kowalski and Sergot in 1986 is a formal system that has evolved from the Situation Calculus (SC). Developed by McCarthy in 1963 the SC was intended to provide an environment for acting agents (Shanahan, 1990). However, instead of using time points for the description of actions and change it only names their result state and thus works merely on frozen states of this world. Each situation is therefore a collection of states which are clearly comparable and separable from the set of states that follow on the next action. In this way, change is modelled in a discrete and strictly ordered manner and hence, partially changing objects or overlapping actions are difficult to represent. Furthermore, the SC has effect axioms to state what changes as a result from an action and it has frame axioms to represent what stays the same. This is necessary to solve the frame problem which was also first recognized by McCarthy and Hayes in 1969.

In comparison to the SC, the Event Calculus (EC) is more flexible and also applicable to problems involving partial change, continuous change and actions that have duration and may overlap with each other. This is achieved by introducing a new sort called *event* to the ontology which represents actions with or without being initiated by an agent. A *fluent*, however, is a familiar type known from the SC, it represents properties in the EC as well as partial changing objects and is typically derived from first-order formulae by a process called reification: the transformation of formulae into terms. While *events* happen, *fluents* hold and can therefore be direct arguments for the truth-predicate. In the EC, a property can also be an argument for another predicate and thus be an object. This is due to the possibility that the state of a partial changing object can, for instance, introduce the existence of that object. For an example see section 5.2.5.

These types, events and fluents, construe the predicates of the language of the Event Calculus. However, there is a third type intermingled with these two and this are the real numbers. The language ($\Re$,0,1,+,*,<) consists of the real numbers and the operations addition and multiplication on them and is used in the EC to represent time points and stages of partial changing objects. It is particularly important to incorporate this time component to ensure that fluents and events can be located temporally and can be extended in time and thus gain a great advantage over the SC. The reals can be axiomatised in a first-order system and surely this is the reason for this choice. Nevertheless it is not clear why real numbers are necessary, in fact, it is hard to imagine a situation which cannot be modelled using integers only. Section 4.3 on continuous change resumes this issue.

3.2 The Axiomatic System

The many-sorted first-order logic called Event Calculus can be sufficiently characterised by few axioms using the following predicates:

1. *HoldsAt(f, t)*
2. *Initially(f)*
3. *Happens(e, t)*
4. *Initiate(e, f, t)*
5. *Terminates(e, f, t)*

1. is a truth-predicate stating that a fluent f holds true at a time point t. Several conditions can evoke this, e.g. when it has been true initially (stated by 2.) and has not been interrupted or terminated. An event type e that happens at time point t is represented by 3. and by being assigned a specific time the event type becomes instantiated and thus an event token. An event may initiate a fluent (at the same time point that it happens at) or terminate it, i.e. an event may cause a fluent to start holding (4.) or to cease holding (5.). It is worth mentioning here that a fluent that is initiated only starts to hold at the following time point (the same applies for terminating it), hence a fluent is interpreted as a set of time intervals of the form *(a,b]* (a is the time instant of the initiating event, so f does not hold yet), and b is the time instant of the terminating event where f still holds). This definition seems to be merely a matter of consistency and homogeneity and it is intuitively clear that one obtains a consistent theory of time without having to switch between types of intervals.

With the five predicates briefly mentioned above it is possible to model instantaneous change, i.e. a notion of causality which describes a sudden change that is induced by an event from one time point to the next. However, there is another notion of causality which necessarily has to be included in a system that claims to model natural language and hence real world propositions. This second notion is continuous change, it is induced by an ongoing action or force that culminates in the result state. This type of change is conceptually very closely related to the aspectual class accomplishment investigated in 5.2.5 ff. For being able to model continuous change as well two more predicates are needed:

6. *Trajectory*(f_1, t, f_2, d)
7. *Releases(e, f, t)*

6. states that if a fluent f_1 holds from time point t to time point t+d then at time point t+d fluent f_2 starts to hold. Thereby the force of a fluent, say an activity, onto another fluent can be expressed. 7. is a predicate that represents the onset of a fluent f by an event e at time point t. To explain why this is necessary we will take a look at the law of inertia proposed by McCarthy and Hayes.

Law of Inertia. *Normally, given any action (or event type) and any fluent, the action doesn't affect the fluent.* [3]

The *Releases*-predicate discards the law of inertia, because an event has 'started' a change and will thus take effect on the fluent continuously. Without the *Releases*-predicate the event in question would only take immediate effect and the fluent would remain in that state. Unlike the *Initiate*-predicate which models only sudden change induced by an event, *Releases* allows a continuous force to have an impact on the fluent as well. The idea is that a continuously exerted force may eventually take effect on a fluent without the presence of an explicitly mentioned event.

The following two predicates are not 'atomic', they just conveniently sum up a couple of conditions that lead to the notion of minimal models (see 3.4).

8. *Clipped*(t_1,f,t_2)
9. *Declipped*(t_1,f,t_2)

The first of these two predicates states that a fluent f does not hold at a time instant t_2 if it held at an earlier time instant t_1 and there has been a terminating (or releasing) event between t_1 and t_2. The second predicate grasps the condition that if a fluent f has been initiated (or released) by an event between time points t_1 and t_2, then f holds at t_2. Hereby events that are *relevant* for the fluent f have been captured and all other events are not taken to have any effect on this fluent.

[3]From (Hamm & van Lambalgen, 2000)

3.3 Scenarios

Scenarios are the kind of system needed to model real world propositions. They are build by introducing states which are defined as follows:

Definition 1. *A state S(t) at time t is a first order formula built from*

1. *Literals of the form ¬HoldsAt(f, t) for t fixed and possibly different f,*
2. *Equalities between fluent terms and between event terms,*
3. *Formulas in the language of the structure ($\Re$,0,1,+,*,<).*

A scenario is on this basis defined as follows.

Definition 2. *A scenario is a conjunction of statements of the form*

1. *Initially(f)*
2. *S(t) → Initiates(e, f, t)*
3. *S(t) → Terminates(e, f, t)*
4. *S(t) → Releases(e, f, t)*
5. *S(t) → Happens(e, t)*
6. *S(f_1, f_2, t, d) → Trajectory(f_1, t, f_2, d)*

where S(t) is a state in the sense of definition 1.

Such a scenario for drawing a circle may look like this:

1. *Initially(nocircle)*
2. *Initially(circle(0))*
3. *HoldsAt(circle(360),t) ∧ HoldsAt(drawing,t) → Happens(complete,t)*
4. *Initiates(start,drawing,t)*
5. *Releases(start,circle(0),t)*
6. *Initiates(complete,existcircle,t)*
7. *Terminates(complete,drawing,t)*
8. *HoldsAt(circle(x),t) → Trajectory(drawing,t,circle(t+d),d)*

Apart from the *Initially*-statements, the above statements are rules that apply at any time, so they are universally quantified over time points. What is still missing now are certain 'facts' in addition to the *Initially*-predicate, for instance, a *Happens*-predicate that will set off the process and release the "time". This only means that the rules will be applied one after the other and in the end one will hopefully obtain the *existcircle*-fluent which is the goal of drawing a circle. Hence,

9. *Happens(start,0)*

Some remarks to creating such a scenario: The reader may have noticed that according to Hamm and van Lambalgen's definition for scenarios and due to the specific notation, the above system already looks like a Prolog program with its distinction between rules and facts where facts are events and fluents with an assigned time point. Furthermore, it should be noted that this is a vast simplification of drawing a circle in the real world. Time, for instance, is suitably represented by real numbers, but for practical reasons in logic programming which is used to 'execute' the Event Calculus, I use integers to represent time steps. Additionally, a function such as circle(0) that is meant to represent the progress of drawing a circle in terms of its already completed degrees is defined in the *Trajectory*-predicate to proceed absolutely uniformly and in linear accordance with each time step.

3.4 Minimal Models

I want to present the idea behind minimal models only briefly to round off the draught of Hamm and van Lambalgen's elaborated System EC and to forestall questions concerning the Frame Problem.

What is usually known in Artificial Intelligence as the Frame Problem is the distinction between things that change from one moment or "frame" to the next and things that stay the same and, additionally, how this can be modelled. Obviously it implies an overwhelming demand on the algorithm to cover all things that stay the same in an environment. Thereby a further distinction is made between the quantitative efficiency of the representation and the computational complexity or in other terms between the Representational Frame Problem and the Inferential Frame Problem respectively (see Russel & Norvig, 2003). The Situation Calculus has aimed at solving this problem by introducing frame axioms in addition to the effect axioms. The former state what remains unchanged while the latter determines the immediate consequences of an event or action performed. This may still be reasonable as long as computations are carried out in an environment like the blocks world which is very restricted and transparent. However, for the modelling of a real world situation this approach would result in a hopeless axiom proliferation while some relevant world knowledge still may not have been considered.

For a sophisticated model of action, time and change which can be applied to more complex domains, one needs a formalisation of reasoning which deals efficiently with the available information but which is flexible enough to readjust its conclusions in case it obtains relevant possibly contradictory information at a later stage. Because classical first-order logic has problems with the formalisation of this type of non-monotonicity the Closed World Assumption (CWA) was introduced. It defines that everything that changes is stated so that whatever does not occur in the database as a fact is simply assumed to be false.

In the EC, the notion of a minimal model relates to that of the CWA in that it is

"characterised by the fact that the occurrences of events and their causal influences are restricted to what is required by the scenario and the axioms of the event calculus." (Hamm & van Lambalgen, 2004)

In other words, all relevant facts and relations are stated and hence, whatever is not stated is assumed to at least have no relevance on the fluents and events of the scenario. The necessity for minimal models lies in the demand on computationability since the concern of the models is the semantics of an expression such as "John draws a circle" and not what could happen in every possible situation such that he never finishes to draw the circle.

Although the intuition about minimal models is clear there are a couple of formal issues that appear problematic. The EC is a formal system that depends on the *HoldsAt*-predicate as truth-predicate. Easily, one can construct natural language expressions with self-reference and this can be done in the EC as well. Self-reference concerning a truth-predicate results in paradoxical situations. Consider the sentence "I am false." where I refers to exactly that sentence. In the EC, this leads to iterated truth-predicates:

$$HoldsAt(F) \leftrightarrow \neg(HoldsAt(HoldsAt(F)))$$

which is contradictive. Those statements are neither true nor false and mainly there are two ways of dealing with this phenomenon:

- One can use a three-valued logic which allows for the value "*neither* true *nor* false" (leads to minimal models) or the value "both - true *and* false" (leads to maximal models).
- The truth-predicate is a partial truth-predicate (T and $\bar{T}$) which only covers true and/or false statements.

Hamm and van Lambalgen use the latter option, just as the theoretic framework given by Feferman in section 5.3 requires it. However, it is obviously problematic to find a minimal model for a statement that can neither be evaluated to true nor to false.

Another problem with minimal model arises when disjunctions are allowed in the EC. If a predicate X is evaluated to true if either a predicate Y holds or another predicate Z holds then there exist two minimal models to this scenario in one of which X is proven by Y and in the other X is proven by Z. Thus, they cannot be compared and are both 'minimal' in parallel. It is unclear which of those two minimal models would then be preferred to represent the reference of the algorithm.

4 CLP and EC

4.1 Non-monotonicity

In 2.3, problems have been described that discourse may bring about when computing the semantics of a sentence. In this sense, natural language is non-monotonic as new information gained at a later stage may alter or even reverse earlier drawn inferences. Obviously, using the Closed-World Assumption does not make sense in this context, hence, an adequate framework for reasoning with natural language needs to have a mechanism for continuously updating the inferences drawn from the changing input. This in turn is an argument for an algorithmic solution which can compute the minimal model that is consistent with the data over and over again while, for instance, the classical model theory is too static to account for this kind of non-monotonicity. Non-monotonicity can result in inconsistencies in such a model when contradicting data is obtained. Then this model can either be modified such that it treats inconsistencies as local without impact on the theory or one has to change the model such that is is rearrange each time an (inconsistent) information is obtained. This is the way how the EC works. The computed minimal model comprises only the given data and assumes everything beyond that to be false or at least irrelevant. All inferences drawn are based on these (minimal) premisses. Hamm and van Lambalgen believe that an algorithm producing such minimal models for an expression is equivalent with the sense or intension of that expression (see 7.1 for details) while the minimal model can be identified with its denotation.

The authors prefer to use Constraint Logic Programming as working environment for the EC to using Prolog for the following reason: planning and therefore interpretation of natural language is closely related to time, as argued in 2, and the EC framework has been developed mainly to be capable of dealing with various temporal reasoning problems. Therefore an implementation of this framework demands a proper representation of time. This is typically achieved by using the reals which were introduced to the EC ontology in 3.1. Since CLP can handle reals and relations between them this solution is consequentially preferred to Prolog which has trouble with reals and cannot resolve constraints of the form $t_1 < t_2$. Moreover, the EC enabled us to equip events and fluents with the potential to be extended in time. Therefore an expression like "John played a sonata" yields a time interval for which 'John played' is true and this interval contains as many time points as the sonata lasts. When asking the program to compute *when* John actually played a Prolog program will return each single time point as a solution for which it holds that John plays. Intuitively this is not what is wanted, a preferred solution would be the presentation of a set of intervals for which the action of playing holds true.

After all, CLP also handles negation in a different way than Prolog does.

In Prolog, negation is achieved by failing to prove the positive counterpart of the negated. Additionally to this, CLP uses constructive negation where answer substitutions are computed. That means, answers to a negative goal can also be computed whereas in Prolog it is impossible to ask e.g. for an X such that ¬p(X) holds. Consider this example by David Chan [4] in Prolog:

```
p(X) : − ¬q(X).
q(X) : − ¬r(X).
r(1).
r(2).
```

The query p(1) succeeds but the query p(x) will fail and not yield the already known answer "1". Constructive negation can handle such queries by establishing new bindings for query variables. If, for instance, $A_1....A_k$ are answers to the query Q then $Q \equiv A_1 \wedge ... \wedge A_k$. If there are no answers, however, the right hand side is empty (i.e. false) which means that ¬Q (negation by failure). The constructive negation rule establishes: $\neg Q \equiv \neg(A_1 \wedge ... \wedge A_k)$. The negation of the answers to Q is returned as answers to ¬Q.

4.2 The Event Calculus in Constraint Logic Programming

In this section, I want to briefly sketch what a CLP algorithm that computes the sense of an expression that is modelled in the EC may look like. First of all, the basic constraint logic program consists of clauses where each has a body and a head. The body in CLP may contain predicates of the EC language as well as numeric constraints while the head must be a primitive predicate of the EC. The general mechanism is that if the body parts can be proven then the head holds true and the overall aim of such a program is to compute a predicate and express it merely as a set of constraints where variables are assigned value as far as that is possible. Such a CLP-clause has the following form

$$B_1, B_2, ..., B_n, c \rightarrow A$$

where B_n and A are primitive EC predicates and c is a numeric constraint typically representing succession of time points. To invoke a computation and ask the program whether a certain goal state can be derived or simply asking it what the denotation of an expression is, a query has to be posed which consists of one or more primitive predicates as well as constraints listed by commas. That is:

$$?c, B_1, ...B_n$$

An example of a simple scenario in CLP can be found in the appendix in section 1. It is taken from (Hamm & van Lambalgen, 2004) and works analogous to the circle-drawing example given above. The program consists of the

[4]From (Henriksson, 2003)

EC axioms and the statements constructing the scenario. Unfortunately the computation cannot be successfully executed on the program as it stands, demonstrated also in section 1 of the appendix. The recursive nature of the predicates leads to infinite loops. I will reconsider this issue in section 4.4.

4.3 Continuous Change in the EC

Although I have already mentioned the reals and their importance for a sophisticated formalisation of reasoning, I still owe an elaboration of how they eventually come into play and help modelling continuous change in the Event Calculus. This is important for modelling a process called auto-termination, i.e. an action represented by a fluent persists over a period of time which eventually causes an event that terminates this action.

An example is the filling of a kitchen sink, where, once the tap-on event has set off the water flow, the sink fills up with water continuously until either the maximum water level has been reached and there happens an overflow or someone turns the tap off and thus terminates the filling process at a lower water level than the maximum. So *Terminates*- and *Initiates*-predicates are crucial in this context. Furthermore, the whole process depends on the filling function which integrates time and the quantitative water level and which determines when and how fast the maximum water level is reached. For a short model of the kitchen sink filling that incorporates continuous change by using integers see the appendix, section 2. However, this example adapted from Shanahan bears problems in the execution as well. The way the program is presented in (Shanahan, 1990) it does not work; as Shanahan admits, some transformations to the code have to be carried out to prevent looping, however he did not suggest a solution. The modifications that I conducted resulted in an executable program that enabled correct computation with a correct result. Nonetheless, the program does not incorporate real numbers, as the title of this subsection suggests. It uses a simple function to increase time and (water)level stepwise. Although this is not truly continuous it suffices to model the situation appropriately and, in fact, it is probably sufficient to model all situations because the real world does not need real numbers.

4.4 Problems

Theoretically the suggestions that Hamm and van Lambalgen make about implementing the Event Calculus in CLP are reasonable and promising. However, it turns out that writing a program in the way a scenario is presented in their book does not lead to a successful computation. After implementing several example scenarios and modifying them in order to make the programs run I conclude that either the EC scenarios have not been tested empirically as CLP programs *or* their implementation in a logic program is not as intuitive and easily inferrable from the (simplified) theoretic presentations given in the book.

The main problem that I encountered when testing the examples were the multiple recursive calls of the *HoldsAt*-predicate that usually led into infinite loops. The axioms of the Event Calculus include four rules for this predicate which in turn are recursive and thus, a failure of unification with an initially holding fluent requires the program to find a different way to prove the current HoldsAt-predicate and ultimatively leads the program into the next (deeper) level of recursion (see 1. in the appendix and the attached commentation on looping of the program).

However, it is possible to write programs which execute the desired computations and yield the expected outcome. Although some familiarities are preserved they do not look like the presented examples in (Hamm & van Lambalgen, 2004) and I find the choice of predicates not quite as intuitive as the authors probably had in mind when choosing CLP as their computational environment for the Event Calculus (see appendix sections 2 and 3).

The main issue here is that the axioms as presented by Hamm and van Lambalgen do not support a CLP implementation. They are not entirely inductive and lead a logic program into infinite loops. Therefore, either the axioms have to be modified to be compatible with CLP or the programs simply do not represent the full theoretic approach because only parts of the axiomatised Event Calculus can be implemented in a running program.

5 Coding Verb Phrases

This section deals with how verb phrases can be represented and coded in the EC. First, an introduction into aspectual classes is given and second, the representation of verbs as eventualities according to their aspectual classes is explained.

5.1 Aktionsarten/Aspectual Classes

According to Vendler [5], verbs can be categorised into four or five different aspectual classes. The distinction is based on the temporal extension and the internal structure and complexity of a verb (phrase). For example, a verb which represents an action that can last over time and has no particular internal structure, i.e. is not implying an starting or ending event, is classified as an *activity.* Here are the five most common and most general aspectual classes (or *Aktionsarten*):

Points: flash

[5](Vendler, 1967)

States: sleep, know
Activities: run, draw
Achievements: reach, arrive
Accomplishments: cross the street, fill a glass

Points had not been included in this list originally but Hamm and van Lambalgen have decided to do so and as the examples above show, the corresponding verbs can somehow be separated from the rest of the classes. However, I think that the distinction between activities and points can be problematic in cases where the former are very short (compare "to kick"). When does an action become punctual and when is it only `short` in duration? And is an achievement not a point as well? In any case the claim is not that every verb belongs to one of those categories in an absolute sense. It is merely the case that a verb or verb phrase can occur with a different aspectual class in a different context. This phenomenon is dealt with in section 6.3.

5.2 Eventualities

An eventuality is a construction that has four parameters which are modified according to the features distinguishing the aspectual classes. In that sense, it represents the internal structure of a verb phrase and consequently, eventualities can be used to determine what has to be included in a corresponding scenario.

Definition 3. *An eventuality is a structure* (f_1, f_2, e, f_3) *where*

1. f_1 *is a fluent which represents an activity, something which exerts a force*
2. f_2 *is a parametrised fluent, representing a parametrised object (partial changing object) or state*
3. e *is the culminating event, representing a canonical goal*
4. f_3 *is a fluent which represents the state of having achieved the goal.*

The sentence "John has build a house." serves as an example for an eventuality structure: The force here is *building*, the partial changing object (subject to the force) is represented by a parametrised fluent *house(x)* which denotes the progress in building a house. The culminating event happens when a constant c is reached that determines that the house has been finished, hence, the *finish*-event depends upon *house(c)* (e.g. *Happens(finish-house,t)*), and finally, the goal state is a state that represents something similar to *exist-house*.

An eventuality also expresses the internal structure that a verb can have. For this reason, an eventuality also determines what has to be included in a scenario to model the specific verb phrase or a proposition containing this verb phrase (VP), as the case may be. Each parameter may occur or not which is indicated by a + and/or a − respectively in each slot of the eventuality structure (force, state, goal event, goal state). Using this representation the aspectual classes look like this:

Point (-,-,+,-)
State (-,-,-,+)
Activity (+,+/-,-,-,)
Achievement (-,-,+,+)
Accomplishment (+,+,+,+)

Whether or not activities contain a state fluent depends on whether one allows for the incremental theme (the change of what is object to the activity) in the representation or not.

The example sentence above serves to illustrate this concept as well. As demonstrated, all eventuality parameters are filled, i.e. they can be unified with items of the sentence, which determines that this sentence expresses an accomplishment.

5.2.1 Points

Points seem to be similar to achievements, they are events which mark one time point and may take effect on fluents and activities by releasing, terminating or initiating them. In contrast to achievements, points have no typical result state as is recorded in its eventuality structure. This type is in the EC represented by the *Happens*-predicate.

5.2.2 States

States and Activities can be very similar and are both represented by fluents in the EC. A formal parameter as given in the eventuality structure is needed to distinguish them in the model. This distinction is based on the notion of causality. While a state is a property that may indeed involve change as exemplified in:

She is even more beautiful today than she was in her younger years.

it is nevertheless defined to be causally inert. Thus, a state can be effect of a cause but cannot actively cause a change in the sense that a force or an event can release fluents. Hence, a state can occur as third but not as first argument in a *Trajectory*-predicate (recall that this argument, if it holds from t to $t+d$, causes the state fluent, the third argument, to hold true at $t+d$ as well). The kind of change illustrated in the example sentence can be modelled by using a graded function representing the state.

5.2.3 Activity

Activities in the wide sense are defined by an activity fluent (in the narrow sense) which is the force that changes an object or a property. The latter is represented by the state of the partial changing object in second argument position of the eventuality structure. The force can also be a graded function since it depends, for instance, on how much energy a runner puts into running and thus, the grade of the force will influence the state fluent, also being graded. Syntactically, an activity can for reasons of cause and volition not be released whereas a state may well be done so. The difference between the two classes seems to lie conceptually in the focus of activity and state. While the change of a state concerns the outcome - what is changed - the profile of an activity lies in the input since that determines the duration and outcome of the activity. Both classes have in common that they are also called durative or atelic because of their lack of a culmination event.

5.2.4 Achievements

Achievements are event types that result from an ongoing activity and terminate it. Thus, an achievement is syntactically given by a goal event which produces a goal state, represented by a *Happens*- and a *HoldsAt*-predicate. Achievements and accomplishment can both be characterised as terminative or telic because they contain a culmination event type.

5.2.5 Accomplishments

Accomplishments have the most complex internal structure and combine all of the above types. They consist of an activity and the state of a partial changing object as well as a terminating event which depends on the stage of the partial changing object and a result state, thereby integrating an activity in the wide sense with its canonical goal if such exists. Compare the elaborated version of an example below to that in 3.3:

1. *Initially(circle(a))*
2. *Initiates(start,draw,t)*
3. *Initiates(finish, circle(c),t)*
4. *Terminates(finish, draw, t)*
5. $HoldsAt(draw,t) \wedge HoldsAt(circle(c),t) \rightarrow Happens(finish,t)$
6. *Releases(start, circle(x), t)*
7. $HoldsAt(circle(x),t) \wedge x=g(s) \wedge y=g(s+d) \wedge s \leq t \rightarrow$
   $Trajectory(draw,t,circle(y),d)$

The activity fluent here is *draw* and the partial changing object is represented by *circle(x)*. The partial changing object *circle* is graded and x marks the corresponding state of the object in degrees. a is the starting stage which may be set to 0 and c is the stage where one considers a circle to be complete, for instance 360 in degrees or 2π in radian. The existence of a circle can be represented by declaring an object *Circle* which depends on *circle(c)*, e.g.: *Circle(circle(360))*.

The *Release*- and the *Trajectory*-rules in the scenario are called dynamics for obvious reasons. They ensure that action takes place in this model while the rest can be more or less considered a database. The complex rule in 7. is necessary to make the partial change dependent on the time interval in that draw is true and not on time itself. If the person is interrupted in drawing from t_1 to t_2 and then proceeds, the *circle*-fluent must be at t_2 at the same stage as it was at t_1.

5.3 Remarks

Now, to represent a natural language expression as a scenario in the EC one needs to construct the eventuality structure. Therefore, the natural language expression has to be analysed and decomposed into fluents and events. To explain this process Hamm and van Lambalgen give a short account of the Feferman calculus which contains a formalisation for verbs as event types and fluents. This calculus was originally developed for a different purpose but suits the needs of this framework. Roughly summarised, an intransitive verb that takes an entity into subject position and a time point can be represented as an event type $\exists t.run(x,t)$ which is an object and as fluent $run[x,\hat{t}]$ which is a function that takes a time point and returns a truth value, each depending on its grammatical use. So, for instance, is the infinitive of `run` an object while an inflected form is a fluent and therefore a function. These distinctions are essential in the formalisation process and become interesting when looking at phenomena involving nominalisation where verbs take on the form (and some of the grammatical behaviour) of a noun. In this study, however, I will not go into detail of those phenomena but rather concentrate on VPs with different aspect and tense. Nevertheless, it is important to note here that the choice of representation of natural language expressions essentially determines the outcome of computations. After all, the mode of representation incorporates lexical and structural information already and, thus, influences the outcome of the model.

6 Progressive and Coercion

Typically, only activities and accomplishments occur in the progressive and achievements or states do not. An achievement is concerned with the result of some activity while the latter is not explicitly mentioned. Although its result fluent can be in a *Trajectory*-predicate, it is lacking the first argument of the *Trajectory*-predicate, the activity. The progressive, however, is concerned with the relation between those two fluents which is characterised by the dynamics

$$S(f_1, f_2, t, d) \rightarrow Trajectory(f_1, t, f_2, d)$$

expressing the state of the two fluents and their relation in the *Trajectory*-predicate. Because an achievement or a state lack either one of those fluents they cannot occur in a progressive form. However, they can be forced into the progressive but then lose their classification as such and become an activity or accomplishment (for details see 6.3).

6.1 The Imperfective Paradox

In linguistics, when concerned with the progressive, the main question typically posed is

> *how the meaning of a sentence using the progressive is related to the meaning of the corresponding nonprogressive.*

Several proposals have been made which I will briefly present in this section. Most of them have led to what is known as the imperfective paradox and by explaining the approaches and their problems, I hope to convey a better understanding of this phenomenon.

One of the first proposals about progressive and its relation to the nonprogressive brought forward by Scott and Montague produced the following definition:

Definition 4. *A simple sentence in the progressive is true at a given time if and only if the corresponding nonprogressive sentence is true at every moment throughout some open interval about t. (Montague, 1974)*

Consider the example sentences "Mary is leaving." and "Mary leaves." and the corresponding past tense expression "Mary has left.". Scott and Montague claimed that the latter sentence holds true at a time point t if and only if the corresponding present tense sentence is true at some time point s where s is before t. But because "Mary is leaving." entails that "Mary leaves." according to the above definition, it also entails that "Mary has left." will be true at some later time point t. This leads to the wrong entailment relation between a progressive sentence and the happening of the culmination event.

The reason why this phenomenon received the name "imperfective" paradox lies in the concept of imperfect and perfect nominalisation. An instance for nominalisation is the gerund which is a nominal built from a verb. A nominal is perfect if it shows the grammatical behaviour of a noun whereas it is imperfective if it is still "verb-like", in fact being a gerund it is still very "progressive-like". Perfective nominalisations have the property of being holistic and can therefore be placed temporally as a whole event, e.g. "The drawing of the circle took place before noon." This is a perfect nominal and the event cannot be further decomposed but located in time, i.e. one has to infer that the process of drawing and the culmination event of finishing the circle took place and that before noon. Opponent to this example, the progressive verb form in "Mary was leaving." does not entail that the culmination event will have taken place. It therefore describes an event that is decomposable into preparatory process and culmination event and, thus, is not 'perfect'.

A second approach suggested to use intervals instead of instants of time and the concluding definition was:

Definition 5. *A simple progressive sentence is true at an interval of time I, if and only if I is a moment of time and there is an interval I' which contains I such that the nonprogressive form of the sentence is true at I'.*

But even with this definition from (Bennett & Partee, 1978) we still get a time point after the interval *I'* where "Mary leaves" was true and at this time point we yield the proposition that "Mary has left." which leads us to the same problem as just described.

David Dowty then introduced a completely different idea based on possible worlds. He suggested that a progressive sentence should be true only if the corresponding nonprogressive was true in all inertia worlds. With 'inertia worlds' he labels worlds that are perfectly similar to the actual one up until the time point in question and that then proceed "normally", i.e. most consistently with the given circumstances. (Note that this notion is comparable to that of a minimal model described in 3.4.) More formally this theory can be described by definition 6.

Definition 6. *PROG(ψ) is true at a pair of an interval and a world <i,w> iff for some interval i' which includes i as a non-final subinterval and for all inertia worlds w ϵ INR (<i,w>), ψ is true at <i',w'>. (Dowty, 1979)*

PROG refers to ψ in the progressive form and *INR* denotes the set of inertia worlds for interval *i* and world *w*. But what happens if the actual world is one of those inertia worlds what may well be the case if it develops in the most compatible way without unforeseen interruptions? Then the nonprogressive holds and we are once again confronted with the imperfective paradox.

Paul Portner has developed this theory in (Portner, 1998) further and combined it with an approach based on events. But essentially he merely elaborates on the concept of inertia worlds by introducing a set of circumstances *(Circ(e))* relevant to whether the "event" (Hamm and van Lambalgen would use the term eventuality here) will be completed, i.e. whether "Mary has left.", and a set of propositions *(NI(e))* which assert that the event does not get interrupted. The set of inertia or 'most compatible' worlds for interval *i* and world *w* are then *Best(Circ, NI, e)* and as such this replaces *INR* in Dowty's definition. 'Most compatible' obviously is a problematic term since it is not formally defined but describes the idea of minimal change from one time point to the next and therefore, is comparable to inertia worlds that proceed "normally". Thereby, it corresponds to the notion of a minimal model which does not allow for unforeseen change. To my mind, the sets *Circ(e)* and *NI(e)* are somewhat unreasonable since they comprise the frame problem without allowing for a solution in the way that minimal models do.

Others have argued that the process of *leaving* or of *drawing a circle* can only be named so if the actors, i.e. the persons that leave or draw, indeed have the *intention* to fulfill the task or to invoke the culmination event. But as Hamm and Lambalgen put it, it is rather difficult to access the intentions of a person which makes it very hard to know initially what kind of process is going on, whether a person is drawing a circle or a square, for instance.

Terence Parsons has discussed those approaches and presented his own idea which is also based on the distinction between events and properties. He proposes the following rule to handle the progressive:

> If 'A' is an event verb, then 'be A-ing' is to be treated semantically as a state verb; otherwise, 'be A-ing' is to be treated the same as 'A'. (Parsons, 1990)

Consequently, he uses in his model a predicate similar to the *Happens*-predicate for event verbs in simple tenses and an equivalent of the *HoldsAt*-predicate for state verbs where the progressive form of an event verb is such a state verb. However, I am convinced that it is not sufficient to represent the progressive using only the state form of the corresponding verb. After all, there is a *default* culmination event that takes place unless something unforeseen happens in the process of the activity.

In the Event Calculus activities are, like states, also represented by the *HoldsAt*-predicate because they are fluents as well. But in contrast to Parsons' approach, an activity is part of a scenario which presents a plan to reach a goal and thus includes the inherent goal as well. This scenario contains universal facts and rules and since it does not use existential quantification it does not enforce the actual occurrence of the goal. In terms of the EC this means that a scenario must only include the goal as event type which is characteristically not instantiated by a time point whereas the scenario does not necessarily have to include the corresponding event token. In other words, the goal is known and lies somewhere in the future and the plan is created to reach the goal but it may not succeed if, in a minimal model, unforeseen events occur and the goal may not actually be reached.

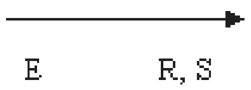


Figure 3: *Event time (E), Relevance time (R) and Speaking time (S) of "I have caught a flu."*

6.2 The Progressive in EC

In this section, I introduce a new concept called "integrity constraints" that helps the Event Calculus to deal with what Reichenbach called "Reference Time" [6]. Contrary to imperfect nominals, perfect nominals can be considered as event *types* for they cannot be punctually located in time, unlike event *tokens*. Accomplishments and activities can be classified as event types. Imperfect events have an internal structure and each of its parts can be located in time and, therefore, they are event tokens. The internal structure contains among other things a reference time which indicates where the relevance of the utterance lies and which may well be different from the event time. An imperfect event may, for instance, contain a starting point for a state or an activity but the relevance of the event may lie somewhere along the time line when the state or activity has been going on for a while. A typical example for this is the English perfect since it usually expresses something about the present using the past to indicate that this 'something' has started some time ago (see figure 3). It is not trivial how to integrate the reference time. In the case of the present perfect where there is an initial fluent holding true we want to emphasise that it is still holding *now* and thus want to infer $HoldsAt(f, t')$ where *t'= now* from the initial situation. If this was just added to the database a simple scenario would therefore include:

Initiates(e,f,t)
HoldsAt(f,now)

However, using axiom 3 to infer from *HoldsAt(f,t')* that an event has happened that initiated f does not work since *HoldsAt(f,t')* can **also** be proven by unifying it with *HoldsAt(f,now)* from the database!

$$[Happens(e,t) \wedge Initiates(e,f,t) \wedge \text{ t < t' } \wedge \neg Clipped(t,f,t')]$$

$$\vee [now = t'] \rightarrow HoldsAt(f,t')$$

This does not yield the desired derivation of *Happens(e,t)*. An integrity constraint can be used to fix this problem. It is a concept borrowed from

[6]From (Hamm & van Lambalgen, 2004)

database theory and it expresses an obligation that the states must satisfy in case they fulfill a condition which invokes the constraint. The syntax of this relation is given by this example from (Hamm & van Lambalgen, 2004):

$$HoldsAt(rain, t) \rightarrow HoldsAt(carry - umbrella, t + \epsilon)$$

The arrow here is meant in an imperative sense that **if** it rains **then** you should carry an umbrella. If this integrity constraint is posed to the logic program and the fact added that *HoldsAt(rain, now)*, where *now* is a real number from the constraint language, the program will try to prove *HoldsAt(carry-umbrella, now +* ϵ*)* (recall that goal and query are equivalent terms in logic programming). More formally an integrity constraint is defined by:

Definition 7. *Let R, R', R" ... be a finite set of constants each denoting a reference time; these constants belong to the constraint language. An integrity constraint is a formula of the form*
(†) IF ϕ *THEN* ψ *(R, R', R"...),*
where ϕ *and* ψ *are formulas of the event calculus.*
The operational meaning of (†) is that if the scenario satisfies ϕ*, the goal ?*ψ*(R, R', R"...) must succeed or fail finitely. To determine whether the scenario satisfies* ϕ*, one has to investigate whether the goal ?*ϕ *succeeds. Hence if the integrity constraint expresses an obligation it may be represented by the demand that ?*ϕ, ψ*(R, R', R"...) succeeds.*

Returning to the example with the flu, instead of adding the fact that *HoldsAt(having-flu,now)*, it is advisable to use the integrity constraint and query

$$?HoldsAt(having - flu, now)$$

which only succeeds if there was a starting event (*Happens(e,t),t <now*) that initiated the flu and the flu has not been healed ($\neg$*Clipped*). This way, the predicate gets extended in time, from the initiating event up to the present and the accent is put on the present state to express the relevance of the perfect such that the person in question is still having a flu at the deictic *now.*

Furthermore, an integrity constraint may not have a condition (or a condition ϕ being a tautology), it must then be achieved in any case and the query solely contains the consequent ψ.

The progressive is only applicable to a dynamic eventuality where there is an activity fluent f_1 and a parametrised fluent f_2 that f_1 takes effect on in the way determined by this equation:

$$HoldsAt(f_2(x), t) \rightarrow Trajectory(f_1, t, f_2(x'), d)$$

where the parameter x depends on the elapsed time. This change is captured by *x'* in the *Trajectory*-predicate. If the eventuality does not itself feature such a dynamics the application of the progressive will coerce (force) the verb phrase into such a structure and extend the scenario accordingly. The phenomenon of coercion is investigated in detail in section 6.3. Assuming then that the scenario exhibits this dynamics, and as such we are dealing with an activity or an accomplishment, the temporal aspect of the present progressive of a verb phrase can be defined by the integrity constraint

$$?HoldsAt(f, R), R = now \tag{1}$$

This ensures that again the starting event has to be computed and the fluent f must not be clipped such that f becomes extended in time which is exactly what the progressive expresses. Otherwise, if one merely adds the fact that f holds now, the event could be only punctual.

The effort of solving the imperfective paradox using the EC leads us back to the notion of minimal models and event types. As has been argued before the scenario representing a proposition is a minimal model in which only events occur that result from the rules and facts of the scenario and that nothing else will influence the course of events.

Theorem 1. *Let $\mathcal{P}$ be the logic program consisting of EC and the scenario given in 5.2.5. Suppose $\mathcal{P}$ is extended to $\mathcal{P}$' so that the query ?HoldsAt(draw, now) succeeds in $\mathcal{P}$'. Suppose $lim_{t\to\infty}g(t) \geq c$. Then comp($\mathcal{P}$') has a unique (minimal) model, and in this model there is a time $t \geq$ now for which HoldsAt(circle(c), t). By virtue of the stipulation that Circle(circle(c)), there will be circle at time t.* [7]

In other terms, the scenario in which *?HoldsAt(draw, now)* succeeds and that therefore represents a present progressive then a minimal model can be computed where no unforeseen events happen (like in an inertia world) and thus it has a predictable future with regard to a default culmination event. Hence, an integrity constraint for this scenario exists that represents the default culmination event as an event type

$$?HoldsAt(circle(360), R), R > now \tag{2}$$

which will be true at R some time in the future if no further *Happens*-predicates that may intervene are added. Adding the universal rule *Terminates(run-out-of-ink,draw,t)* is an event type as well and will not influence the goal computation as long as *Happens(run-out-of-ink,t_1)* $\wedge$ *($g(t_1) <$ 360)* has not been added. By adding this *Happens*-predicate, we experience a non-monotonic change and the logic program will then compute that the culmination event does not take place, i.e. (2) fails and hence, the expression has a different meaning.

[7]Adapted from (Hamm & van Lambalgen, 2004), a proof for this theorem can be found on p. 160 pp

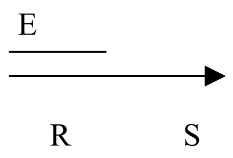


Figure 4: *"John was building a house." : Relevance time (R) lies inside Event (E), Speaking time (S) is 'now'*

The past progressive is similar to the present progressive concerning the temporal aspect. The difference is that the reference time, like the event time, lies in the past although the activity may still last (to the deictic *now*) as in "John was building a house". Here the focus is on the event in the past but one does not know whether he finished it or whether he is still building (figure 4). The past progressive can thus be represented by the integrity constraint and query

$$?HoldsAt(f, R), R \leq now \tag{3}$$

which starts a computation similarly to that of (1).

6.3 Coercion

Verbs can be classified in terms of aspectual classes but the classes have no explanatory power and the verbs do not belong to one class exclusively. The verbs rather change depending on the object noun phrase they occur with or an adverbial construction that may take influence. "Writing" is an activity, for instance, while "writing ones bachelor's thesis" is an accomplishment because the object inherently brings a culmination event with it that is determined by the logical configuration of the *thesis.* Accordingly, the aspectual class changes when a person "plays the piano" or "plays a sonata". Furthermore adverbial constructions such as "for 2 hours" in combination with another subject can induce a change. Coercion is one view upon this shift of aspect which is considered to depend on a force (context) that imposes a temporal structure upon a verb and overrides the 'default' aspectual class. Consider the achievement verb *arrive.*

a. John arrived.
b.*John arrived all night.

Here 'arrived' as an achievement cannot be extended with the durative adverbial construction. However,

c. Guests arrived all night.

allows for a durative sense and it may look like it has become an activity at first glance. But I object to this view adopted by Hamm and van Lambalgen (Hamm & van Lambalgen, 2004, p. 213) because the difference to the first sentence lies solely in the plural noun phrase in subject position that enables a reading which quantifies the achievement such that it applies for each guest individually. Nevertheless, syntactically it behaves like an activity because of its characterisation via a durative fluent and as such it would have to be represented by a *HoldsAt*-predicate. Another phenomenon which also fits under the heading 'coercion' is where presumingly stative verbs become activity verbs under the influence of the progressive.

a. I love her.
b.*I am loving her.
c. I am loving her more and more each day, the more I get to know her.

Love in its simple form can clearly be classified as a stative verb because it denotes a property, a feeling. The progressive can then, in the right context, impose an internal structure on the verb such that it changes from being an atom in a. to having stages and phases in c.

In the following sections, I outline the three types of coercion that Hamm and van Lambalgen distinguish and sketch the formal EC treatment for the individual phenomena.

6.3.1 Additive Coercion

Additive Coercion is a label for those phenomena where something is added to the scenario to obtain the 'new' aspectual class. Consider an activity like 'drink' and the accomplishment 'drink a glass of beer'. The scenario for 'He drinks.', for instance, is given by *HoldsAt(drink, t)*. It has to be extended with a dynamics which enables reaching the culmination event and the direct object which specifies the culmination event to yield the new scenario:

1. *Initially(glass(full))*
2. *HoldsAt(drink, t)* $\wedge$ *HoldsAt(glass(empty), t)*
   $\rightarrow$ *Happens(finish-glass, t)*
3. *Terminates(finish-glass, drinking, t)*
4. *HoldsAt(glass(x), s)* $\wedge$ $x = g(s) \wedge y = g(s{+}d) \wedge s \leq t$
   $\rightarrow$ *Trajectory(drink, t, glass(y), d)*

Of course the appropriate *Initiates*- and *Releases*-predicates have to be added to complete this scenario but I think the idea has been conveyed. Similarly does the transformation from achievement to accomplishment work. Opponent to the former case does an achievement provide the culmination event but lack the preparatory phase, the activity, and the parametrised fluent which leads to the culmination event. Hence the scenario must be extended with a dynamics like above introducing the appropriate activity fluents.

6.3.2 Subtractive Coercion

This type of coercion is contrastive to additive coercion and applies mainly to accomplishments which have been turned into an activity by removing their culmination event, e.g. when an interrupting event prevents culmination such that it cannot take place the accomplishment remains in the activity phase. In the same way that 'drink' became an accomplishment by adding 'a glass of beer' these predicates can be then eliminated to once again obtain the scenario for 'drink'. For an EC example see the *crossing*-scenario in section 3 of the appendix.

6.3.3 Cross-Coercion

This seems to be the most interesting class of coercion phenomena that Hamm and van Lambalgen scrutinise. Some elements are added while others have to be deleted to derive the coerced meaning. They distinguish and elaborate mainly on three subtypes of cross-coercion: state $\rightarrow$ activity, structure vs phenomenal interpretation and point $\rightarrow$ activity of which I will present the first and the last type only for reasons of relevance to this thesis.

State $\rightarrow$ activity

The second example of the introduction to this section on Coercion serves as an example for the transformation from states to activities. 'I love her' can be represented by the integrity constraint *?HoldsAt(f, R), R = now* with *f* representing the stative verb 'love'. The progressive and the addition of "more and more each day" coerce this verb into an activity such that love as state is represented by a fluent *f* and a parametrised fluent *f'* is introduced to capture the rise of intensity of loving. Furthermore the progressive imports a dynamics which is familiar by now and abstractly looks like this:

1. *Releases(e, f_2, t)*
2. *HoldsAt($f_2(x)$, t) $\rightarrow$ Trajectory(f_1, t, $f_2(x')$, d)*

Hamm and van Lambalgen suggest that the variables f_1 and f_2 in the dynamics are unified with *f* and *f'* respectively while the starting event initiating *f* is computed by the integrity constraint. This account seems to produce nice results but is this not due to the rather deliberate creation of two distinct fluents from one verb namely to love? I postpone this issue to section 6.4 and finish this part with another possibility of representing the 'love'-coercion. One may object that not 'loving' is the activity in this sentence but 'getting to know her' which does not change anything but it requires that f_1 and f_2 are simultaneously substituted by *g* and *g'(x)* respectively with *g* denoting the state fluent 'know' and *g'(x)* denoting the parametrised fluent 'know *to a degree*'. *g'(x)* is then related to the parametrised fluent loving *f'(x)* by the following rule:

$$HoldsAt(g'(x), t) \rightarrow HoldsAt(f'(x), t)$$

Point → activity

The coercion from a point event to an activity is marked by the difference in the time span. Obviously, a point event is a punctual event which occurs at one instant only. If a temporal phrase like "for two hours" is added then it becomes an activity and can only be interpreted to have an iterative reading.

a. The light flashed.
b. The light was flashing for two hours.

But how can such an activity be represented as a fluent? The suggested solution is based on the definition of fluents as functions from time points to truth values. So the fluent '*flash(for two hours)*' must be a function over events that takes time points and returns a truth value. *Happens*[*flash,* $\hat{t}$] is such a fluent in the frame work of the Event Calculus. In the scenario this fluent would be identified with the partial changing object because it is limited by the adverbial construction "for two hours" and it progresses while time goes on. This identification with the appropriate variable in the *Trajectory*-predicate is achieved via unification. Moreover, an activity with a partial changing object needs a dynamics and a driving force, the actual activity, and for this Hamm and van Lambalgen choose an *unspecified* activity. Unfortunately, I cannot conceive of this idea as reasonable in a formal system like the EC since it seems impossible for the EC to correctly produce something like an unspecified activity. Even though if it was added manually it would not make sense intuitively.

6.4 Discussion

It is without doubt a difficult venture to construct a theory that accounts for the phenomenon of coercion. A number of people have been concerned with change of aspect and its rules. Alessandro Zucchi is one of them, in (Zucchi, 1998) he also investigates this phenomenon but under the more general label of 'aspect shift' for which lexical ambiguity and coercion pose different approaches in explaining it. Zucchi points out that only some verbs allow aspect shift while others do not and he expects to find many answers in finding the rules that distinguish those verbs from each other. On the basis of work done by Barbara Partee, he proposes that stative verbs like 'love' must allow to be put into a pseudo-cleft construction ("What I did was...") as a precondition for occurring in the progressive. So "I love her" can be transformed to "What I did was (to) love her." and therefore the progressive form "I am loving her more and more each day" also exists. Unfortunately, I have not found an answer to the question why "What she did was resemble her mother." is ungrammatical but "She is resembling her mother more and more each day" works fine.

It turns out that all of the accounts that have been mentioned in this thesis exhibit some weaknesses that many open questions remain which leaves a lot of space for ideas and suggestions. I find the event approach that Hamm and Lambalgen follow very sensible and I hope to have shown that it can handle many difficult cases convincingly. However, doubts remain whether single phenomena have been assessed correctly. Consider again the example of the introduction to this section, here conveniently repeated in a whole sentence:

(1)
a. John plays the piano for two hours.
b. John plays a sonata for two hours.

The NP in a. does not contain a definite internal length so the verb denotes an activity which can go on arbitrarily without any default culmination event. On the other hand in b. we combine that same verb with an object that determines the length of the play because it is itself defined to last only a certain interval of time. It follows that we obtain an iterative interpretation in all those cases where the individual temporal length of an object is smaller than that of the adverbial phrase. This shows that subjects and objects have a strong impact on aspect and therefore need a proper representation. Now, the iterative reading consists in a repetition of accomplishments since the sonata is played several times, in fact it is played (*given time span* divided by *interval for one sonata*) times. But how can that be formalised and automated, how can this activity have the internal structure of many accomplishments? The second example from the introduction confronts the EC with a similar problem.

(2)
a. John arrived.
b. Guests arrived all night.

Here in (2) b. we encounter something that looks like an activity because of the time interval over which it stretches. But 'arrive' is a punctual event, an achievement, which does not *hold* all night, rather many such instants where someone arrives occur induced by the plural subject. So what is modified by the temporal specification is not the event of arriving but the time span in which 'arrive' happens, still punctually. So once again, as in (1) we gain an iterative reading through the additional adverbial phrase. In my opinion, the solution to (2) as presented in the section on point→activity does lead into the right direction, although I find it problematic to add an unspecified activity to such a scenario because these irregularities make the automation of such processes very difficult.

Another difficulty pose examples of accomplishment verb phrases whose activity stages are lexically so tightly bound to the culminating event that they do not allow for a coercion to an activity, in other words their culmination event cannot be prevented. This is the case with verbs like 'to die' or 'to persuade' as in (3).

(3)

a. *John persuaded me to come but in the end he didn't succeed.

b. ?He was dying but after one week treatment with the new medicine he recovered.

Hamm and van Lambalgen do not hesitate to classify those examples as accomplishments and agree with Comrie who asserts that here culmination event and preceding activity cannot be separated. Their explanation on why those verbs are different is that for b. there exists no plan to reach the final event and as such the person in question cannot actively interfere with what is going on. Hence, a solution would be to introduce different sorts for events (natural, self-initiated and initiated by others) to constrain the terminating event to the process with the condition that it may not be self-initiated. Similarly, they suggest that a constraint stating that only the default culminating event can terminate the process in b. be a solution to that phenomenon. I agree with the analysis of these two examples and I admit that the suggested solutions may work but once again they seem extraordinarily arbitrary and I doubt that the information that is needed to integrate such constraints into a scenario is lexically accessible.

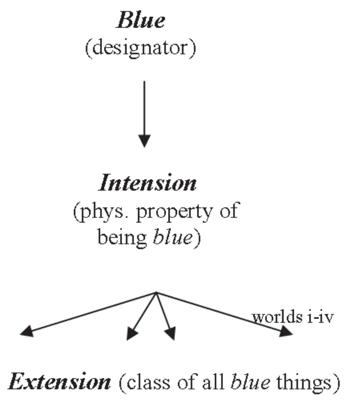
7 Coercion and Intensionality

Hamm and van Lambalgen's higher goal in modelling this algorithmic framework lies in developing an explanatory theory that accounts for what Frege called 'Sinn' and 'Bedeutung', in English meaning or sense and denotation or reference. They aim at finding a more suitable representation of those concepts that lead to many problems with so-called 'intensional' constructions. I explain and discuss this issue in the following subsections, starting with a short introduction to Frege and Carnap in which I make heavy use of the books (von Kutschera, 1989) and (Krauth, 1970).

7.1 Sense vs Intension

Gottlob Frege was one of the most important personalities, if not the founder, of modern language philosophy. In his work on 'Sinn und Bedeutung' he reasoned about the designator (or 'Zeichen') and the designated (das 'Bezeichnete'). Frege said that a name or designator *means* its corresponding denotation in the real world, where meaning is the translation of Frege's "Bedeutung" and as such is a technical term which is equivalent to 'denotation', 'reference' or 'extension'. However, he points out that the difference in statements is not solely dependent on the difference between their meanings otherwise "a = a" were equivalent with "a = b" given that $a = b$ is true. Rather the second statement bears some information that the first does not, provided that the designator *'b'* differs in a relevant way from the name *'a'*. Thus, he suggests that the two identity statements have to be distinguished if the difference in names corresponds to a difference in the mode of the presentations ('Art des Gegebenseins') of the designated. The famous example of the morning star / evening star illustrates this principle. Therefore, to a designator Frege relates a denotation in the real world as well as a 'sense' of this designator which expresses the mode of presentation. The sense can be a proper name or the description of a geometrical phenomenon and the meaning is what is named or described. Further, he reasons that a designator can be meaningless, i.e. have no extension, but has a sense in the way that "Odysseus" or "the smallest real number greater than one" have. Frege defines the sense as something that presents a certain feature of the designated, a relevant information that contributes to a complete characterisation of the designated. One problem with this is that it is not clear in what way proper names contribute such an information and hence what their sense is. The sense of designators can be tested for identity if the designators can be exchanged *salve veritate* that is without changing the meaning of the expression.

Sentences and predicates need particular elaboration since they are more complex and their meaning cannot be an object or a number. For sentences Frege suggests that their meaning be computed from the meaning of its parts and the sense be computed accordingly. These are the two functional principles, in linguistics also known as "principle of compositionality", which are the bases for modern computational semantics. But what is the meaning of a sentence? Frege correctly reasons that the proposition ('Gedanke') of the sentence changes when parts with identical denotations but different senses are substituted and therefore the proposition cannot be the meaning. What remains the same even after such a substitution are the truth conditions of the sentence, what has to be the case for the sentence to be true. Hence, he identifies the meaning of a sentence with its truth value and the proposition it expresses with its sense. This explains why statements of the form "a = b" and "a = a" differ given that $a = b$ is true: since the meaning of "a" and "b" is identical we know by applying the functionality principles that the meaning of the two statements (sentences) is the same. But because the sense of "a" may be different from that of "b" the sense of "a = b" may also be different from that of "a = b". The sense is somewhere between the designator and the designated, on a logical level, which allows for the sentence to be evaluated logically disregarding whether it is true realistically.



Blue
(designator)
Intension
(phys. property of being blue)
worlds i-iv
Extension (class of all blue things)

Figure 5: *The intension of* BLUE *determines its extension in all possible worlds*

One of the dominating theories in linguistics competing with Frege's proposal has been suggested by Carnap, who was one of Frege's scholars. He essentially adapts this distinction between designator and the two dimensions of the designatum where one is the **extension** in the real world and the other what Carnap calls the **intension** and which is roughly equivalent to Frege's sense. Further, Carnap introduces the possible world idea which has been mentioned in a modified from in this thesis when presenting Dowty's account for the progressive in section 6.1. The extension is defined by Carnap to mean not only the denotation in the real world but to subsume the denotations in all possible worlds. A possible world for him is nothing else than a possible set of circumstances which are could be the case instead. The intension is, similarly to the sense, a property of the designated that serves to identify the extension and it is also on a logical level between the designator and the real world's and possible worlds' denotations. To exemplify the three concepts consider figure 5.

Carnap agrees with Frege on the extension of a sentence to be its truth value and the intension to be the proposition expressed. Similarly, he proposes that expressions are extensionally identical if they are factually equivalent and intensionally identical if they are logically equivalent and therefore have the same extension in every possible world (e.g. "7" and "seven" express exactly the same property and have the same extension). The intension can, in combination with all facts, determine the extension which is stated again below by definition 8.

Definition 8. *The intension of an expression A is the function that maps to each possible world the extension of A in that world.* [8]

[8]Translated from (von Kutschera, 1976)

More formally this could be expressed by a formula $I(A) =< w_i, E_i(A) >$ where w is a world and E the corresponding extension of A.

Although there have been different proposals about defining intension and extension, Carnap's approach is probably one the most significant theories in the history of linguistics. For this reason, his conceptualisation has been used for a comparison to the algorithmic approach to intension and extension that is presented in this study. The comparison and discussion are carried out in the subsequent sections.

7.2 Coercion and Intension

The claim that Hamm and van Lambalgen assert in their book states that coercion is an intensional phenomenon which they support by pointing out the analogy to the sense/meaning levels of names and predicates. Accordingly, an event has no canonical referent or meaning, rather it is constructed from a sense. Following Frege, the name of an object bears a sense that characterises the object in a particular way. In the same way can an algorithm be considered to be the sense that characterises an event and as such the meaning of that event (expression) can be computed by the algorithm just as the extension of a name/predicate can be determined by means of its intension (and all facts). In the case of the algorithmic equivalent the scenario specifies the "Art des Gegebenseins" by incorporating universally quantified rules corresponding to the sense and the facts of the situation which makes it possible to compute the minimal model which in turn is the reference of the expression. Coercion is supposed to exemplify this theory since a subsequential natural language remark may transform an accomplishment to an activity by eliminating the culminating event and therefore the expression must be re-analysed to obtain the new denotation. The analogue to this constant mental re-analysis is the re-computation by the algorithm that permanently processes new incoming data which may lead to a modification of the sense. The consequence that the authors draw from these considerations is to state that the algorithmic approach is not only an analogue but in fact a reasonable way to model how we conceive and deal with sense and reference as an alternative way to possible worlds. Therefore, they pose the controversial claim that:

> The sense of an expression is the algorithm which computes its reference.

If the sense is the algorithm then coercion is a process that maps one "sense of an expression to a different but related sense of [that] expression" via unification as has been suggested in section 6.3. This statement follows naturally from the above assumption and that becomes more obvious if one reconsiders an often cited example of a progressive sentence "John was crossing the street." which possesses a scenario representing its meaning.

When a subsequent clause introduces an intervening event which terminates the crossing before the natural culmination event (truck hits John) then that expression has a modified sense represented by the extended scenario which in turn computes the denotation, i.e. the minimal model in which John does not reach the other side of the road.

7.3 Discussion

Opponent to possible world semantics the idea outlined in this thesis has the demand of being cognitively relevant. As shown in section 2 and in the descriptions in 7.2, the authors justify their approach by pointing out the similarity to human processing of language, supported by empirical evidence. On the other hand, the main concern of possible world semantics is in determining the content of the representation of the truth conditions for an expression instead of claiming cognitive relevance at all (Wilson & Keil, 1999).

This algorithm-based approach seems sensible to me and it is a great combination of mathematical approved methods with linguistic investigation techniques to find an appropriate model for language which is of impact for many areas since sense and denotation are relevant not only for natural languages but for formal languages as well. A weakness could be assigned to the question whether this framework is realistically applicable at all to natural languages since the world knowledge and the human intuition that is required in so many situations to construe a scenario in the first place is vast. In some way, the problem of vagueness in the possible world semantics has been shifted to a problem of creating the appropriate scenario which is explicit and therefore very sensitive to lexical and syntactic variations. Furthermore, the transformation from language to the Event Calculus ontology has not been comprehensibly presented. It may be the case that this is being thought of as secondary but considering the amount of examples that illustrate practical application details it must be of interest to the authors. And finally, it is not quite clear whether the claim that sense is an abstract algorithm that computes the reference is meant to have explanatory power for mental processes or whether the model is but highly suitable to simulate the latter functionally.

8 Conclusion

The approach presented by Hamm and van Lambalgen is a useful and reasonable alternative to possible world semantics. It states that an abstract algorithm represents the sense of a term or program respectively. What is calculated by this algorithm corresponds to the reference of this term or program. In the context of the Event Calculus this reference is the minimal model. Whilst possible world semantics may equip one with a certain ability to understand the notion of sense and develop an intuition about distinguishing sense and reference of an expression (or intension and extension) respectively, it does not possess a cognitive justification. In Cognitive Science the goal in all disciplines is, after all, to investigate cognitive phenomena and discover and use cognitive processes. In that way the algorithmic approach has been a fine piece of scientific work which proceeded in adapting a system, the Event Calculus, from Artificial Intelligence, substantiating it with mathematical reasoning and projecting it to the domain of linguistics while using logic programming to realise this construct.

However, several issues need further investigation. It is, for instance, not clear how natural language expressions are correctly represented by the Event Calculus. So far, this step has been a rather arbitrary process which is largely based on intuition. Moreover, it has not been trivial, in fact, to construe a CLP program from the scenarios given by Hamm and van Lambalgen as can be seen in the appendix, section 1. Nevertheless working versions can be found in the appendix and although they do not exactly correspond to the theoretic instructions on how to construe a scenario, the programs support the core idea. The sense of an expression can be modelled by such an algorithm and, using the notion of minimal models, the predicted outcome is indeed correct as in the example of the progressive expression "crossing the street" where the street will have been crossed at some stage. If the following expression or sentence includes, however, an event that interferes with this prediction then this simply has to be added to the existing scenario. To this extend, the original idea of related senses represented by related algorithms can be supported and nicely illustrated. However, it is important to note, that the manual formulation of the predicates is driven by what is expected to be the outcome. In that way, it is not particularly surprising that the programs work in the desired manner. Moreover, the axioms of the EC should be reconsidered in view of the problems arising with their implementation. After all, CLP is the computational environment chosen by Hamm and van Lambalgen to implement the EC and therefore, the theoretic framework should be somewhat more consistent with its computational application.

Nevertheless, Hamm and van Lambalgen's idea is sensible and I endorse their very elaborated framework which may not be of practical relevance for modelling natural language (yet) but it may certainly be used to model parts. Furthermore, because of their limited lexical diversity and their restricted domain, formal/programming languages could well be a realistic application.

References

Bennett, M., & Partee, B. (1978). *Toward the Logic of Tense and Aspect in English.* Bloomington: Indiana University Linguistics Club, revised and extended version of 1972 system development corporation (santa monica, california) report edition.

Dowty, D. (1979). *Word Meaning and Montague Grammar.* Reidel, Dordrecht.

Hamm, F., & van Lambalgen, M. (2000). Event calculus, nominalisation and the progressive. *Linguistics and Philosophy*, 76 pp.

Hamm, F., & van Lambalgen, M. (2003a). Intensionality and coercion. *Intensionality.*

Hamm, F., & van Lambalgen, M. (2003b). Moschovakis' notion of meaning as applied to linguistics. *Logic Colloqium '01.*

Hamm, F., & van Lambalgen, M. (2004). *The Proper Treatment of Events.* Blackwell.

Henriksson, J. (2003). *Briefly on constructive negation.* Linköping University.

Krauth, L. (1970). *Die Philosophie Carnaps.* Library of Exact Philosophy. Springer Verlag.

Mariott, M., & Stuckey, P. (1998). *Programming with Constraints: An Introduction.* MIT Press, Cambridge, MA.

Montague, R. (1974). *Formal Philosophy: Selected Papers of Richard Montague.* Yale University Press, New Haven.

Moschovakis, Y. (2003). *A logical calculus of meaning and synonymy.* A corrected and edited version of a set of notes for a course in NASSLLI '03.

Parsons, T. (1990). *Events in the Semantics of English. A Study in Subatomic Semantics.* MIT Press, Cambridge, MA.

Portner, P. (1998). The progressive in modal semantics. *Language*, *74*(4), 760–87.

Russel, S., & Norvig, P. (2003). *Artificial Intelligence. A Modern Approach.* Prentice Hall.

Shanahan, M. (1990). Representing continuous change in the event calculus. *Proceedings ECAI* (pp. 598–603).

Shanhan, M. (1997). Event calculus planning revisited. *Proceedings 4th European Conference on Plannning '97* (pp. 390–402).

Tschorn, P. (2001). *Prolog - Programming in Logic.* University of Osnabrück, ISIV.

Vendler, Z. (1967). In *Linguistics in Philosophy*. Cornell University Press, Ithaca, New York.

von Kutschera, F. (1976). *Einfuehrung in die Intensionale Semantik*. Roland Posner, De-Gruyter-Studienbuch.

von Kutschera, F. (1989). *Gottlob Frege. Eine Einführung in sein Werk*. De-Gruyter-Studienbuch.

Wilson, R., & Keil, F. (Eds.). (1999). *The MIT Encyclopedia of Cognitive Sciences*. MIT Press, Cambridge, MA.

Yeom, J. (2003). The semantics of the english progressive and the imperfective paradox. *Language and Information*, *7*, 139–161.

Zucchi, A. (1998). In S. Rothstein (Ed.), *Events and grammar*, Studies in Linguistics and Philosophy (pp. 349–370). Kluwer Academic Publishers, Dordrecht.

APPENDIX

1 - Example Scenario for ”crossing the street”

Statements

```
initially(one-side).
initially(distance(0)).

happens(reach,T) :-
    holdsAt(distance(M),T),
    holdsAt(crossing,T).

initiates(start,crossing,T).
initiates(reach,other-side,T).

releases(start,distance(0),T).

terminates(reach,crossing,T).

trajectory(crossing,T,distance(X+D),D) :-
    holdsAt(distance(X),T).
```

Axioms

```
holdsAt(F,0) :-
    initially(F).
holdsAt(F,T) :-
    holdsAt(F,R),
    R < T,
    not(clipped(R,F,T)).
holdsAt(F,T2) :-
    happens(E,T1),
    initiates(E,F,T1),
    T1 < T2,
    not(clipped(T1,F,T2)).
holdsAt(F2,T2) :-
    happens(E,T1),
    initiates(E,F1,T1),
    T1 < T2,
    T2 is T1 + D,
    trajectory(F1,T1,F2,D),
    not(clipped(T1,F1,T2)).
```

```
clipped(T1,F,T2) :-
    happens(E,S),
    T1 < S,
    S < T2,
    (terminates(E,F,S);
    releases(E,F,S)).
```

Failure due to recursion

?holdsAt(other-side,T)

```
(1) 1 CALL holdsAt(other - side, T)
(2) 2 CALL initially(other - side)
(2) 2 FAIL initially(...)
(1) 1 NEXT holdsAt(other - side, T)
(3) 2 CALL holdsAt(other - side, R)
(4) 3 CALL initially(other - side)
(4) 3 FAIL initially(...)
(3) 2 NEXT holdsAt(other - side, R)
(5) 3 CALL holdsAt(other - side, R)
(6) 4 CALL initially(other - side)
(6) 4 FAIL initially(...)
(5) 3 NEXT holdsAt(other - side, R)
(7) 4 CALL holdsAt(other - side, R)
(8) 5 CALL initially(other - side)
(8) 5 FAIL initially(...)
(7) 4 NEXT holdsAt(other - side, R)
.
.
.
```

This documents the execution of the program and illustrates the infinite looping that it gets stuck in. It was reported by the tracer of the ECLiPSe program that I chose for executing the CLP files. The program consists of the EC axioms and the statements that Hamm and van Lambalgen provide to model the specific scenario for "crossing the street". Obviously, this mode of presentation is not suitable to enable the desired computation.

2 - Continuous change: the "kitchen sink" example

```
outlet(8).
outlet(10).

releases(start,tap-on,0).
releases(endfilling,tap-off,T).

:- dynamic(holdsAt2/2).

holdsAt2(level(0),0).

holdsAt(level(L),T) :-
    holdsAt2(level(L),T).
holdsAt(level(L),T) :-
    happens(E,T1),
    initiates(E,filling,T1),
    holdsAt2(level(L1),T1),
    count(L1,T1,L,T).

initiates(E,filling,T) :-
    releases(E,tap-on,T).

terminates(E,filling,T) :-
    happens(E,T),
    releases(E,tap-off,T).

happens(start,0).
happens(endfilling,T) :-
    holdsAt2(level(L),T),
    outlet(L).

count(L1,T1,L,T) :-
    holdsAt2(level(L),T).
count(L1,T1,L,T) :-
    not(terminates(E,filling,T1)),
    Lneu is (L1 + 2),
    Tneu is (T1 + 1),
    asserta(holdsAt2(level(Lneu),Tneu)),
    count(Lneu,Tneu,L,T).
```

In my program, some modifications had to be made to the EC axioms as presented by Hamm and van Lambalgen as well as to some predicates in particular.

The *holdsAt*-predicate has been split into two predicates, one representing the (reduced) axioms and the other one providing the recursion base for the induction. The *count*-predicate replaces the *Clipped*-predicate in the Event Calculus. It counts down each time step from the goal back to an initiating event for the fluent in question. Thereby it checks at each time instant whether a terminating event has taken place that may have *clipped* the corresponding fluent. *Asserta* adds to the database that the parametrised fluent *level(X)* holds at the calculated time instant and thus, this information can be used for further processing.

3 - Subtractive Coercion: ”crossing the street” revisited

```
releases(start,distance(0),0).
releases(reach,otherside,T).

initiates(E,crossing,T) :-
    releases(E,distance(X),T).

:- dynamic(holdsAt2/2).

holdsAt2(distance(0),0).

holdsAt(distance(L),T) :-
    holdsAt2(distance(L),T).
holdsAt(distance(L),T) :-
    happens(E,T1),
    initiates(E,crossing,T1),
    holdsAt2(distance(L1),T1),
    count(L1,T1,L,T).
holdsAt(crossing,T) :-
    Tpre is (T-1),
    happens(E,Tpre),
    initiates(E,crossing,Tpre).
holdsAt(crossing,T):-
    Tneu is (T-1),
    not(terminates(E,crossing,Tneu)),
    holdsAt(distance(L),T),
    holdsAt(crossing,Tneu).

terminates(E,crossing,T) :-
    happens(E,T),
    releases(E,otherside,T).
terminates(E,crossing,T):-
    happens(hitbytruck,T).

happens(start,0).
happens(reach,T) :-
    holdsAt2(distance(L),T),
    L is 10.
happens(hitbytruck,3).

count(L1,T1,L,T) :-
```

```
    holdsAt2(distance(L),T).
count(L1,T1,L,T) :-
    not(terminates(E,crossing,T1)),
    Lneu is (L1 + 2),
    Tneu is (T1 + 1),
    asserta(holdsAt2(distance(Lneu),Tneu)),
    count(Lneu,Tneu,L,T).
```

This piece of code models the expression "crossing the street", where the distance from one side of the street to the other is 10 m and 2m are taken at each time step. The street has been crossed if the distance of 10 m has been covered. This terminates the fluent *crossing* and *releases* the distance. The culmination event which is the natural terminating event is represented by reaching the maximum distance. The query *?holdsAt(distance(10),5), holdsAt(crossing,5)* is evaluated to true while *?holdsAt(distance(12),6), holdsAt(crossing,6)* is evaluated to false since the culmination event occurs at time instant 5 and thus, fluents that are terminated by this event cease to hold true from the next time instant on. The addition of an expression "when he/she was hit by truck" to "crossing the street" leads to the addition of the two predicates given above in italics. Adding those two predicates leads to a termination of the *crossing*-fluent at time instant 3 already such that the covered distance does not exceed 6 m.

Eidesstattliche Erklärung

Hiermit erkläre ich, Maria Staudte, die vorliegende Arbeit "Modelling the Progressive and Coercion using the Event Calculus" selbständig verfasst zu haben und keine anderen Quellen oder Hilfsmittel als die angegebenen verwendet zu haben.

Osnabrück, September 2004

Maria Staudte
Matrikel-Nr: 904908