Generating models for temporal representations

Patrick Blackburn, Sébastien Hinderer

Generating models for temporal representations

Patrick Blackburn
INRIA Lorraine
615 rue du Jardin Botanique
54602 Villers ls Nancy Cedex, France
Patrick.Blackburn@loria.fr

Sébastien Hinderer
INRIA Lorraine
615 rue du Jardin Botanique
54602 Villers ls Nancy Cedex, France
Sebastien.Hinderer@loria.fr

Abstract

We discuss the use of model building for temporal representations. We chose Polish to illustrate our discussion because it has an interesting aspectual system, but the points we wish to make are not language specific. Rather, our goal is to develop theoretical and computational tools for temporal model building tasks in computational semantics. To this end, we present a first-order theory of time and events which is rich enough to capture interesting semantic distinctions, and an algorithm which takes minimal models for first-order theories and systematically attempts to "perturb" their temporal component to provide non-minimal, but semantically significant, models.

Keywords

model building, first-order logic, higher-order logic, computational semantics, events, tense, aspect

1 Introduction

In this paper we discuss the use of model building for temporal representations. We chose Polish to illustrate the main points because (in common with other Slavic languages) it has an interesting aspectual system, but the main ideas are not language specific. Rather, our goal is to provide theoretical and computational tools for temporal model building tasks. To this end, we present a first-order theory of time and events which is rich enough to capture interesting semantic distinctions, and an algorithm which takes minimal models for first-order theories and systematically attempts to "perturb" their temporal component to provide non-minimal, but semantically significant, models.

The work has been implemented in a modified version of the Curt architecture. This architecture was developed by Blackburn and Bos [2] to illustrate the interplay of logical techniques useful in computational semantics. Roughly speaking, the Curt architecture consists of a representation component (which implements key ideas of Montague semantics [10]) and an inference component. In this paper we have used a modified version of the representation component (based on an external tool called Nessie written by Sébastien Hinderer) which enables us to specify temporal representations using a higher-order logic called TY_4. However, although we shall briefly discuss how we build our temporal representations, the main focus of the paper is on the other half of the Curt architecture, namely the inference component.

Inference is often though of simply as theorem proving. However one of the main points made in [2] is that a wider perspective is needed: theorem proving should be systematically coupled with model building and the Curt architecture does this. Model building takes a logical representation of a sentence and attempts to build a model for it; to put it informally, it attempts to return a simple picture of the world in which that formula is true. This has a number of uses. For example, as is emphasized in [2], model building provides a useful positive test for consistency; if a model for a sentence can be built, then that sentence is consistent (this can be useful to know, as it enables us to prevent a theorem prover fruitlessly searching for a proof of inconsistency). Moreover, in subsequent papers, Johan Bos and his co-workers have demonstrated that model building can be a practical tool in various applications (see for example [6, 5, 4]).

The work described here attempts to develop a Curt style architecture rich enough to handle natural language temporal phenomena. So far we have concentrated on the semantic problems raised by tense and aspect. We have developed a first-order theory of time and events, which draws on ideas from both [9] and [3]. Although these theories were developed for English, we believe the underlying ideas are more general, and to lend support to this claim we shall work here with Polish.

As we shall see, however, more than a theory of time and events is required. Model builders typically build the smallest models possible, but such models may not be suitable for all tense and aspectual combinations, which often underspecify the temporal profile of the situations of interest. We thus provide an algorithm which takes as input a first-order theory, a first-order formula, and a model for the theory and formula, and systematically attempts to "perturb" the temporal part of the model to find non-minimal but semantically relevant models.

2 Modelling tense and aspect

In this section, we shall discuss the logical modeling of tense and aspect, drawing on some simple examples from Polish, and informally introduce a temporal ontology of time and events which will let us express temporal and aspectual distinctions in a precise way. The formal definition of a theory over this temporal

ontology (which draws on ideas from [3] and [9]) will be given in Section 4.

Consider the following four Polish sentences:

1. Piotr pospaceruje
2. Piotr pokochal Aline
3. Piotr napisal list
   and
   Piotr popisal list

The first sentence refers to a walking event and adopts a perfective point of view: it insists on the fact that the mentioned action will be terminated at some point in the future. The second sentence mentions an eventuality of loving and also adopts a perfective point of view. However, the reading of this sentence differs from the previous one. The first sentence insisted on the *termination* of the event, whereas the second one insists on its *beginning*. In other words, the second sentence has an *inchoative* reading. This is because the verb "kocha" from which "pokochac" is derived is a *state verb*, and perfective state verbs have inchoative readings in Polish. So the second sentence means that at some point in the past Piotr started to love Alina.

The last two sentences, which are also perfective, both refer to the termination of a writing event which is located in the past. The difference between these two sentences concerns the way the writing event terminated. In the "napisac" variant, an idea of successful termination is conveyed: that is, at some point the writing stopped, because the letter was finished. In the "popisal" variant, the writing also stopped but the conveyed idea is that the writing event was interrupted before its normal termination, which implies that the letter could not be finished. To distinguish between a "normal" termination and a termination due to an unexpected, premature interruption, we talk about *culminations*. An event *culminates* when it terminates and has also been completed, or fully accomplished. Thus the event of writing reported by the sentence "Piort napisal list" culminates, whereas the one in "Piotr popisal list" does not.

Note that in our two first examples, it makes no sense to talk about the culmination of the walking or loving eventualities; neither walking events nor states of loving have natural culminations in the way that writing events do. More generally, different types of events have different properties, and verbs can be classified according to the properties of the event they refer to. Such a classification has been proposed for Polish verbs by Młynarczyk [8], and we follow this classification in our work. The classification proposes five verb classes, including the three just mentioned: a class for processes ("to walk" belongs to this class), a class of state verbs and gradual transitions (a member of which is "to love") and a class for culminations ("to write" belongs to this class). Processes are non-instantaneous events which have no particular properties; it is possible to look at them either as ongoing (imperfective), or as finished (perfective). State verbs are also non instantaneous. Their imperfective use corresponds to a vision of the state as holding, whereas (as was already mentioned) their perfective use has an inchoative reading. Culminations have an imperfective variant and two perfective ones: one for events that have culminated, another for event that have not culminated.

Now, our aim is to translate simple Polish sentences like those just discussed into logical formulas that encode their meaning. More precisely, we are interested in obtaining logical formulas that give an account of the sentence's temporal and aspectual properties suitable for theorem proving and model building purposes. This means we should choose a logic that makes it easy to distinguish various kinds of entities (for example, ordinary individuals and events) and that lends itself naturally to semantic construction. To achieve these goals we will use a higher-order typed logic called TY_4. This logic belongs to the TY_n family of logics. This family of logics has long been advocated by Muskens (see, for example, [11]) as an appropriate logical setting for natural language semantics. The four basic vocabulary types we shall build the formulas of this logic over (in addition to the type of truth-values which is always included in TY_n theories) are:

entity : for individuals and objects;

time : for moments of time;

event : for the events introduced by verbs;

kind : to classify events into kinds.

The first type (entity) will certainly be familiar to the reader used to Montague-style semantic construction. The second type, time, is clearly needed to give an account of notions like past, present and future. The abstract entities known as events (introduced by [7]) are a convenient object one can use to talk about actions introduced by verbs. Each verb introduces an event, which is then used to record additional information about the action the verb describes. For example, if the verb "to eat" introduces an event e, then the fact that the entity doing the eating is x will be encoded as $\textsc{Agent}(e, x)$, the fact that the eaten entity is y will be encoded as $\textsc{Patient}(e, y)$, and so on. Event-based representations for the verbs make it easy to attach additional information, for example information contributed by verb modifiers; for each modifier, one simply introduces a binary predicate whose first argument is the event of interest and whose second argument is the piece of information to be attached to this event. Finally, every event has a kind, and we assume that each verb picks out a distinct kind of event.

The logic we work with makes use of the following binary predicates relating events and times:

- $\mathit{inception}(e, t)$ means that the event e starts to take place at the moment t;
- $\mathit{conc}(e, t)$ means that the event e ends at the moment t;
- $\textsc{Induration}(e, t)$ means that the event e is going on at the moment t;
- $\textsc{Ek}(e, k)$ means that the event e is of kind k.

In addition, it has the following binary relation which relates times:

- $\mathit{lt}(t, t')$ means that time t is before time t'.

Furthermore, it has the following binary relation between events:

- $culm(e, e')$ means that event e' is the culmination of event e.

This relation plays a key role in analysing the semantics of verbs like "napisal/popisal".

There are also number of other unary relations involving events (such as $culminated(e)$), and a temporal constant Now to represent the time of utterance. The way these items are inter-related will be formally spelt out in Section 4.

3 Computing semantic representations

Before turning to the formal specification of the theory of time and events, we shall briefly outline the process that allows us to automatically translate Polish sentences into a logical formula over the vocabulary introduced in the previous section. This process is done in three steps: parsing, computing a semantic representation in higher-order logic, and translating this representation to plain first-order logic. The translation process uses a modified version of the Curt architecture.

3.1 Parsing

The parsing is done using a Prolog DCG. It parses the text given as input and produces a syntax tree reflecting its structure. The leaves of this tree can be labelled either by a word and its syntactic category, or by an operator encoding a verb's temporal and aspectual meaning.

For example, here is the parse tree produced for the sentence "Piotr pospaceruje" (Piotr will have walked):

```
binary(s,
       unary(np, leaf(piotr, pn)),
       binary(vp, leaf(pastiv, op),
              leaf(pospacerowac, iv))
      )
```

The first and third leaves refer to lexical entries, whereas the second carries an operator. This operator indicates that the verb carried by the following leaf is in the past.

3.2 Computing higher-order logic representations

This step is performed by an external tool that has been especially developed to compute semantic representations from a parse tree. The tool is called Nessie, and it takes as input a parse tree similar to the one just presented and a lexicon specifying the semantic representation for each word; it was designed to handle the TY_n family of logics. Thus for present purposes we simply declare to Nessie the four basic vocabulary types we have selected (namely entity, time, event, and kind) and Nessie is then equipped to handle the higher-order language they give rise to. The simply typed lambda-calculus lies at the heart of the TY_n family of logics, and Nessie handles such tasks as type-checking and β-reduction. In other words, the work Nessie does is very much inspired by Musken's adaptation of Montague's original approach to natural language semantics.

The output of this second step of processing is, generally speaking, a typed lambda-term. In our temporal representations, once Nessie has β-reduced the term, there will be neither applications nor abstractions present in the final formula. In other words, the semantic formula provided by this second step is close to a genuine first-order formula, the only difference being that the variables occurring in the term are typed.

To continue with our example, Nessie would compute the following representation for the sentence:

$$\exists t : \mathrm{T}ime.\exists e : \mathrm{E}vent.(\mathrm{L}t(\mathrm{N}ow, t) \wedge \mathrm{E}k(e, \mathrm{S}pacerowac) \wedge \mathrm{A}gent(e, \mathrm{P}iotr) \wedge \mathrm{C}onc(e, t)).$$

3.3 From higher-order to first-order representations

In logical semantics there are important trade-offs between higher-order and first-order logics. As Montague, Muskens and others have demonstrated, higher-order logics are a natural medium for specifying semantic theories: their expressivity allows semantic representations for all syntactic categories to be given (and entailment relations between them to be stated). Moreover, the fact that they incorporate the simply typed lambda calculus gives a uniform and simple approach to semantic construction.

But higher-order approaches have a drawback. They are inherently more complex than first-order approaches. Because of this, relatively few automated reasoning tools exist for higher-order logics, and those that do are not particularly efficient. But all is not lost. As formal semanticists have long known, in natural language semantics, the higher-order constructs typically produce representations which are very close to first-order ones. So, if we could translate the TY_n expressions output by Nessie into first-order logic, we could have the best of both worlds.

At first glance, it could seem that the only thing to do to convert a higher-order formula (like the one shown above) into a first-order one is to remove the types. In fact, things are slightly more complex than this, as the following example should make clear. Consider the formula: $\Phi = \forall X : \tau P(x)$, where τ is a type. If we throw types away too quickly, we get as candidate for a first-order translation of Φ: $\Phi' = \forall X P(X)$. But Φ and Φ' don't have the same meaning: the former formula states that the predicate P holds for every object of type τ, whereas the latter claims that P holds for every object, no matter what its type is.

A semantically correct translation can however be obtained, with the help of a unary predicate that characterizes the object of type τ. With the help of such a predicate (which will be written τ'), it becomes possible to propose a semantically correct translation of Φ in first-order logic: $\Phi'' = \forall X(\tau'(X) \rightarrow P(X))$. To obtain a complete specification of a translation function translating higher-order formulas into firs-order formulas, a similar trick should be used for the

existential quantifier: $\exists X : \tau P(X)$ is translated to $\exists X(\tau'(X) \wedge P(X))$. The translation of other formulas is straightforward.

The complete translation mechanism has been implemented in Nessie which can on demand produce either higher-order or first-order semantic representations. Thus, here is the final first-order representation we get for our initial sentence:

$$\exists t(\textsc{time}(t) \wedge \exists e(\textsc{event}(e) \wedge \textsc{lt}(\textsc{now}, t) \wedge \textsc{ek}(e, \textsc{spacerowac}) \wedge \textsc{agent}(e, \textsc{piotr}) \wedge \textsc{conc}(e, t))).$$

4 A first-order theory of time and events

We are interested in computationally modeling tense and aspectual distinctions. In particular, we want to derive logical representations useful for model building purposes. But we have not yet achieved this goal. Although Nessie can output first-order representations, simply giving such representations to a first-order model builder won't give us what we want, for as yet we have said nothing about how the various symbols we are using are interrelated. For example, the previous representation talks about an event taking place in the future, as the $\textsc{lt}(\textsc{now}, t)$ conjunct makes clear. A model for such a representation should of course reflect this. But nothing in the representation itself prevents the model builder from identifying t with NOW, or from building a model where both $now < t$ and $t < now$ hold, as we have said nothing about the properties of NOW or LT or how they are related. And this is only the tip of the iceberg. It is relatively clear what properties LT should have (for example, it should be transitive) but many other constraints (notably on the way times and events are interrelated) need to be expressed too. In short: to automatically compute models for a semantic representation, we need to work with respect to a theory of time and events, and the purpose of this section is to sketch the theory we use.

In essence, the theory we need should take into account some basic typing facts (for example that two objects of different types can not be identified, and that predicates impose typing constraints over their arguments), structural properties of time (such as the transitivity of LT), and, most importantly of all, the way times and events are inter-related. The following sections give first-order axioms which formalise the required constraints. We won't give all the axioms (for example, we omit all axioms covering events for verb classes not discussed here) but we have given enough to convey a flavour of what is required to carry out model building for tense and aspectual information.

4.1 Type definitions

The following axioms state that the set of elements of the models should be partitioned by the four types we use: event, kind, time and entity. The following two axioms are typical:

not_event_entity : $\forall A \neg(\textsc{event}(A) \wedge \textsc{entity}(A))$

not_entity_time : $\forall A \neg(\textsc{entity}(A) \wedge \textsc{time}(A))$

There is also an axiom stating that every object should belong to at least one type.

4.2 Typing constraints

Another family of axioms reflects the typing constraints imposed by the predicates over their arguments. For example, the binary predicate AGENT requires that its first argument is an event and that its second argument is an entity. The following is a sample of such axioms:

now_type:
$\textsc{time}(\textsc{now})$

lt_type:
$\forall A \forall B(\textsc{lt}(A,B) \rightarrow (\textsc{time}(A) \wedge \textsc{time}(B)))$

agent_type:
$\forall A \forall B(\textsc{agent}(A,B) \rightarrow (\textsc{event}(A) \wedge \textsc{entity}(B)))$

conc_type:
$\forall A \forall B(\textsc{conc}(A,B) \rightarrow (\textsc{event}(A) \wedge \textsc{time}(B)))$

inception_type:
$\forall A \forall B(\textsc{inception}(A,B) \rightarrow (\textsc{event}(A) \wedge \textsc{time}(B)))$

ek_type:
$\forall A \forall B(\textsc{ek}(A,B) \rightarrow (\textsc{event}(A) \wedge \textsc{kind}(B)))$

4.3 Structure of time

The previous two groups of axioms were essentially organisational: they laid out the basic constraints individuating types and imposed restrictions and requirements on the relations the various types of entity could enter into. We are now ready to turn to more substantial axioms, that is, axioms that impose structure on our ontology. The simplest such axioms are those regulating the temporal part of the ontology. The following requirements are standard (see for example [1]):

lt_irreflexive:
$\forall A \neg \textsc{lt}(A,A)$

lt_transitive:
$\forall A \forall B \forall C((\textsc{lt}(A,B) \wedge \textsc{lt}(B,C)) \rightarrow \textsc{lt}(A,C))$

lt_total:
$\forall A \forall B((\textsc{time}(A) \wedge \textsc{time}(B)) \rightarrow (\textsc{lt}(A,B) \vee (\textsc{eq}(A,B) \vee \textsc{lt}(B,A))))$

Other axioms could be imposed (such as the requirement that every point has a successor, or that the structure of time is dense) but for present purposes we won't make use of such options. Instead we will turn to the heart of our formalisation, namely its treatment of events and the way they interact with time. This part draws on and generalises ideas presented in [3] and [9].

4.4 Structure of events

This group of axioms is itself divided into three parts, namely general axioms regulating the relationship between times and events, axioms for instantaneous events, and axioms for culminations (actually, in the full version of the theory there are axioms constraining the events required for other verb classes, but we omit them here).

4.4.1 Relating times and events

The following is a sample of the axioms we use to regulate the interplay between the structure of time and the structure of events. As a rough mental picture, it may be useful to think of events as hanging from the temporal structure (a bit like balloons hanging by string from a long stick). The following axioms (which have been abstracted from [3]) then ensure that the two kinds of entity are properly coordinated:

agent_unique:
$\forall A \forall B \forall C((\text{AGENT}(A,B) \wedge \text{AGENT}(A,C)) \rightarrow \text{EQ}(B,C))$

event_has_inception:
$\forall A(\text{EVENT}(A) \rightarrow \exists B \text{INCEPTION}(A,B))$

inception_unique:
$\forall A \forall B \forall C((\text{INCEPTION}(A,B) \wedge \text{INCEPTION}(A,C)) \rightarrow \text{EQ}(B,C))$

event_has_conc:
$\forall A(\text{EVENT}(A) \rightarrow \exists B \text{CONC}(A,B))$

conc_unique:
$\forall A \forall B \forall C((\text{CONC}(A,B) \wedge \text{CONC}(A,C)) \rightarrow \text{EQ}(B,C))$

inception_not_after_conc:
$\forall A \forall B \forall C((\text{INCEPTION}(A,B) \wedge \text{CONC}(A,C)) \rightarrow \neg \text{LT}(C,B))$

duration_before_conc:
$\forall A \forall B \forall C((\text{INDURATION}(A,B) \wedge \text{CONC}(A,C)) \rightarrow \text{LT}(B,C))$

not_inception_and_induration:
$\forall A \forall B \neg(\text{INCEPTION}(A,B) \wedge \text{INDURATION}(A,B))$

not_induration_and_conc:
$\forall A \forall B \neg(\text{INDURATION}(A,B) \wedge \text{CONC}(A,B))$

4.4.2 Instantaneous events

Our account of the semantics of culmination (which is essential for some Polish verbs) makes use of the notion of instantaneous events. There are a number of plausible ways of axiomatising this notion. For model building purposes, we work with the following axioms:

instantaneous_definition_1:
$\forall A(\text{INSTANTANEOUS}(A) \rightarrow \exists B(\text{INCEPTION}(A,B) \wedge \text{CONC}(A,B)))$

instantaneous_definition_2:
$\forall A \forall B(\text{EVENT}(A) \rightarrow ((\text{INCEPTION}(A,B) \wedge \text{CONC}(A,B)) \rightarrow \text{INSTANTANEOUS}(A)))$

Note that the second axiom is the converse of the first.

4.4.3 Culminations

We turn to the semantics of culmination. In essence, this part of our theory formalises key ideas from Moens and Steedman [9]. That is, we view eventualities such as writing a book as a relation between *two* events. The first event is the lead-up, or preparatory process, for example the act of writing. The second event (which we view as instantaneous) is the event of culmination, in the case the event of finishing the book. Sometimes the culmination is not achieved, and Moens and Steedman use evocative terminology to describe what goes on in this case: they talk of the eventuality being "stripped" of its culmination. To use their terminology, Polish lexicalises the distinction between stripped (for example "popisal") and unstripped (for example "napisal") eventualities. The following axioms capture these ideas in a form suitable for model building:

culm_unique:
$\forall A \forall B \forall C((\text{CULM}(A,B) \wedge \text{CULM}(A,C)) \rightarrow \text{EQ}(B,C))$

culm_injective:
$\forall A \forall B \forall C((\text{CULM}(A,C) \wedge \text{CULM}(B,C)) \rightarrow \text{EQ}(A,B))$

culm_no_fixpoint:
$\forall A \neg \text{CULM}(A,A)$

culm_antisymmetric:
$\forall A \forall B(\text{CULM}(A,B) \rightarrow \neg \text{CULM}(B,A))$

culm_preserves_agent:
$\forall A \forall B \forall C((\text{CULM}(A,B) \wedge \text{AGENT}(A,C)) \rightarrow \text{AGENT}(B,C))$

culm_preserves_patient:
$\forall A \forall B \forall C((\text{CULM}(A,B) \wedge \text{PATIENT}(A,C)) \rightarrow \text{PATIENT}(B,C))$

culm_preserves_kind:
$\forall A \forall B \forall C((\text{CULM}(A,B) \wedge \text{EK}(A,C)) \rightarrow \text{EK}(B,C))$

culm_inception:
$\forall A \forall B \forall C((\text{CULM}(A,B) \wedge \text{CONC}(A,C)) \rightarrow \text{INCEPTION}(B,C))$

culm_imp_instantaneous:
$\forall A \forall B(\text{CULM}(A,B) \rightarrow \text{INSTANTANEOUS}(B))$

culminated_definition:
$\forall A(\text{CULMINATED}(A) \rightarrow \exists B(\text{EVENT}(B) \wedge \text{CULM}(A,B)))$

culminated_imp_not_instantaneous:
$\forall A(\text{CULMINATED}(A) \rightarrow \neg \text{INSTANTANEOUS}(A))$

4.5 A first model

With the help of the previously given axioms, a model builder will generate far more reasonable models than the one mentioned at the beginning of this section. As an example, here is the model produced by the `Paradox` model builder for the sentence "Piotr pospaceruje" (Piotr will have walked):

```
D=[d1,d2,d3,d4,d5]
f(0, spacerowac, d2)
f(0, piotr, d1)
f(0, now, d5)          f(2, inception, [(d3,d5)])
f(1, entity, [d1])     f(2, ek, [(d3,d2)])
f(1, event, [d3])      f(2, lt, [(d5,d4)])
f(1, kind, [d2])       f(2, agent, [(d3,d1)])
f(1, process, [d2])    f(2, conc, [(d3,d4)])
f(1, time, [d4,d5])
f(1, instantaneous, [])
```

Roughly speaking, this model describes a situation where Piotr starts to walk right now and finishes its walk at some point in the future.

5 Building non-minimal models

Although the situation described in the model we just built is realistic, it is not the only realistic situation the sentence describes. It is compatible with the semantics of Polish perfective verbs in the present tense that Piotr has already walked for a long time, or that his walk has not started yet but will start later in the future. That is, this particular combination of tense and aspectual information underspecify the temporal profile of the situations of interest.

However model builders typically will *not* find these other models. Why not? Because they are not *minimal*. Model builder attempt to find the smallest model they can, and in the above example it has identified `d5` with both `now` and with the inception of event `d3`. This gives rise to a perfectly legitimate model — but the strategy of identifying points when possible rules out the other two semantic options just mentioned. The other model are non-minimal because they do *not* identify the time of utterance with the inception time. And one of these models may well turn out to be the one required for processing subsequent sentences.

So we need to do more, and this section presents an algorithm which returns a list of *all* the realistic situations, as far as tense and aspect are concerned. The input of this algorithm is a model similar to the one shown in the previous section. The output models can be seen as perturbations of the initial one. The construction procedure takes place in two steps. First, a generation step produces a list of possible models. Second, a selection step is used to filter out those models that actually satisfy both the initial semantic representation and the axioms. The second step essentially uses first-order model checking as described in [2], so we focus here on the generation step.

Our initial input are a sentence S, its representation R as a first-order formula, and a theory T of time and events (such as the one given in the previous section). The formula R is supposed closed and consistent with T. Thus, there is a model M_0 of T in which R is satisfiable. Our purpose is, starting from M_0, to build the set $\mathcal{M}_f$ of all non-isomorphic "minimal perturbations" of models of T in which R is satisfiable.

First, we build a set $\mathcal{M}_i$ of candidate models. All the generated models can be seen as perturbations of the initial model M_0. The part of M_0 that is not related to time and events will be the same for all the produced models. The variations from model to model only affect the points denoting moments in time and relations those points belong to. To put it more precisely, the constant part of the final models (which will be called the *core* in the rest of this paper), is obtained by removing the time-related information from M_0. For instance, if M_0 is the model given previously, then its core is:

```
D=[d1,d2,d3]
f(0, piotr, d1)          f(1, entity, [d1])
f(0, spacerowac, d2)     f(1, event, [d3])
f(2, agent, [(d3,d1)])   f(1, kind, [d2])
f(2, ek, [(d3,d2)])      f(1, instantaneous, [])
                         f(1, process, [d2])
```

From the core model, we build another intermediate model, where all the significant moments in time are represented by distinct points. By significant moment, we mean those moments where something happens. We start by adding a point which interprets the constant NOW. Then, we go through the events present in the core model, and for every event e we proceed as follows:

1. If e is instantaneous, one point d_k is added, and the pair (e, d_k) is added to the INCEPTION and CONC binary relations;
2. If e is not instantaneous, we examine the relations INCEPTION, ınduration and CONC of the model M_0. For each of these binary relations R in which e is involved, we add a new point d_i and extend the relation R of the currently built model with the pair (e, d_i).

Applying this algorithm to the core seen previously yields the following intermediate model:

```
D=[d1,d2,d3,d4,d5,d6]
f(0, piotr, d1)          f(1, entity, [d1])
f(0, spacerowac, d2)     f(1, event, [d3])
f(0, now, d4)            f(1, instantaneous, [])
f(2, ek, [(d3,d2)])      f(1, kind, [d2])
f(2, conc, [(d3,d6)])    f(1, process, [d2])
f(2, agent, [(d3,d1)]) f(1, time, [d4,d5,d6])
f(2, inception, [(d3,d5)])
```

The model obtained after this extension step is quasi-complete. The only missing part is the LT relation specifying how the moments just introduced are ordered. What we do is that we generate all the possible orders (called successions) and, for each succession, we build the associated model.

The number of possible successions grows exponentially with the considered number of moments: 2 moments x and y give 3 possible successions ($x < y$,

$x = y$, $y < x$), 3 moments give 13 successions, 4 moments give 75 successions.

Before a succession is used to complete a model, it is simplified. The simplification consists in replacing all the elements that denote the same moment in time by one single element. For example, the succession $d_i = d_j$ would be replaced by a single element d_k, and a mapping would be generated to rename both d_i and d_j to d_k. This substitution must of course be applied to the intermediate model so that the merges are taken into account correctly.

What we get as result of the succession simplification process is a list of moments in time, and a substitution to be applied to the intermediate model. The order of the elements in the list encodes there chronological order. The final model corresponding to one given succession is hence obtained from the intermediate model by performing the two following steps:

1. Apply the substitution provided by the succession's simplification;

2. If $x_1, \ldots, x_n$ is the list of moments returned by the succession's simplification, every pair (x_i, x_j) such that $1 < i < j < n$ is added to the LT relation. This ensures that the properties of LT such as its transitivity and irreflexivity will hold in the new model.

This marks the end of the first (generation) step we mentioned before. Since the intermediate model we presented before makes use of 3 moments in time, we obtain 13 possible successions, hence 13 possible models. This 13 models are tested (using a first-order model checker) to see which really satisfy both the semantic representation and the theory T. Finally, three models are kept. The first is the initial model M_0 The second looks like this:

```
D=[d1,d2,d3,d4,d5,d6]
f(0, piotr, d1)          f(1, entity, [d1])
f(0, spacerowac, d2)     f(1, event, [d3])
f(0, now, d4)            f(1, instantaneous, [])
f(2, ek, [(d3,d2)])      f(1, kind, [d2])
f(2, agent, [(d3,d1)]) f(1, process, [d2])
f(2, conc, [(d3,d6)])  f(1, time, [d4,d5,d6])
f(2, inception, [(d3,d5)])
f(2, lt, [(d5,d4),(d5,d6),(d4,d6)])
```

As required, this corresponds to a situation where the walking event starts in the past. The third model differs from the second only in the information on the temporal ordering, which looks like this:

```
f(2, lt, [(d4,d5),(d4,d6),(d5,d6)])
```

In this model the walking event starts in the future.

The algorithm also finds the possible models for the other example sentences we talked about in section 2. For the sentence "Piotr pokochal Aline", the system provides the three distinct models. On the other hand "Piotr napisal list" and "Piotr popisal list" only have one model each. The external model builder finds this model, and our algorithm correctly concludes that the model cannot be perturbed.

6 Conclusion

In this paper we have discussed a logic-based approach to modeling temporal information, and in particular, information about tense and aspect. Our approach has been general and generic. On the representational side, we have used a tool called Nessie which allows us to specify temporal (and other ontologies) within the generous expressive limits provided by TY_n. On the inference side we have provided a first-order theory which, although inspired by work on English, seems general enough to provide analyses of tense and aspect in other languages. Finally, we have provided an algorithm which allows us to perturb the temporal component of models in the hope of finding non-minimal but semantically significant variants. This algorithm is not dependent on the axiomatic choices made in this paper; in fact (as we have discovered) is a very useful tool when one is investigating the effects that varying the underlying theory can have.

Much remains to be done. For a start, the work reported here does not consider many other important temporal phenomena, such as dates, temporal prepositions, and temporal adverbs. Furthermore, it is not integrated with a theory of discourse structure; incorporating the ideas reported here into a Discourse Representation Theory (DRT) based approach would be a natural path to investigate. We plan to turn to such extensions shortly.

References

[1] J. Benthem. *The Logic of Time*. Kluwer Academic Publishers, Dordrecht, second edition, 1991.

[2] P. Blackburn and J. Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI, 2005.

[3] P. Blackburn, C. Gardent, and M. de Rijke. Back and forth through time and events. In *Proceedings of the Ninth Amsterdam Colloquium*, pages 161–175, 1993.

[4] J. Bos and K. Markert. Recognising textual entailment with robust logical inference. In J. Q. et al, editor, *MLCW 2005*, volume LNAI 3944, pages 404–426, 2006.

[5] J. Bos and T. Oka. Meaningful conversation with mobile robots. *Advanced Robotics*, 21(2):209–232, 2007.

[6] J. Bos and T. Oka. A spoken language interface with a mobile robot. *Artificial Life and Robotics*, 11(1):42–47, 2007.

[7] D. Davidson. The logical form of action sentences. In N. Rescher, editor, *The Logic of Decision and Action*. University of Pittsburgh Press, 1976.

[8] A. Młynarczyk. *Aspectual Pairing in Polish*. PhD thesis, University of Utrecht, 2004. LOT Dissertation Series 87.

[9] M. Moens and J. Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14:15–28, 1988.

[10] R. Montague. *Formal Philosophy. Selected Papers of Richard Montague. Edited and with an Introduction by Richmond H. Thomason*. Yale University Press, New Haven, 1974.

[11] R. Muskens. *Meaning and Partiality*. Studies in Logic, Language and Information. CSLI Publications, 1996.