PENG Light meets the Event Calculus

Rolf Schwitter

Centre for Language Technology, Macquarie University
Sydney 2109 NSW, Australia
Rolf.Schwitter@mq.edu.au

Abstract. In this paper I present an extension of the controlled natural language PENG Light that is motivated by the formal properties of the Event Calculus, a narrative-based formal language for events, fluents, and the periods for which these fluents hold. I show how this extension can be used to express domain-specific axioms in a natural way and sketch how PENG Light and the Event Calculus can be brought together for reasoning about events and time.

1 Introduction

Machine-processable controlled natural languages can be used as high-level knowledge representation languages that balance the disadvantages of full natural languages and formal languages [2]. These controlled natural languages look seemingly informal since they are subsets of natural languages and are designed in such a way that they can be translated unambiguously into a formal target language (and sometimes vice versa). Taking a logic-based formalism for representing events and change as a target language, I investigate how an existing controlled natural language can be extended in a systematic way in order to specify knowledge about events and their effects including the time periods during which these effects persist. This results in a specification and query language that can be used for updating and interrogating a sequence of unfolding events.

2 PENG Light

PENG Light is a controlled natural language that can be used for knowledge representation and specification purposes [12]. In a nutshell, a PENG Light text consists of a sequence of anaphorically linked sentences where prepositional phrases in adjunct position modify the verbal event. Let us assume a scenario where we can observe and describe the following sequence of events as they unfold:

1. John arrives with Flight AZ1777 at the airport of Palermo at 10:10. John gets on a bus at the airport. The bus leaves the airport at 11:30 and arrives at the port of Trapani at 13:05. John gets off the bus in Trapani.

The temporal structure of these sentences is simple in the sense that temporal expressions can only occur as modifiers of events and that the events in these sentences have a linear temporal order. Apart from describing events, PENG Light can also be used to speak about states, for example:

2. The weather is bad. The wind is strong and the sea is rough.

and to make conditional statements such as:

3. If the weather is good then John boards the hydroplane at 14:10 and arrives on Marettimo Island at 15:35. If the weather is bad then John stays in Trapani and goes to the Albergo Maccotta.

PENG Light sentences can be translated incrementally during the writing process via discourse representation structures [3] into TPTP's first-order form syntax [11]. A Satchmo-style model builder [5, 6] is used to generate a finite model and questions can then be answered over the resulting knowledge base [12], for example questions such as:

4. When does John arrive at the airport? Who gets off the bus in Trapani?

Note that in this case no additional background knowledge is required to answer these questions since the relevant facts can be looked up in the resulting knowledge base. The situation, however, is different for questions such as:

5. Where is John now? Where is John at 13:30? Why does John stay in Trapani?

In contrast to a machine, a human observer can easily infer from the text and his or her commonsense knowledge that John is on the bus at 12:00 but not anymore at the airport of Palermo at that time and that John is in Trapani at 13:30 but not anymore on the bus. One possible way of thinking about this problem is that if an event happens at a specific time point then this event initiates a state that holds after that time point and continues to hold until another event terminates this state. Thus, the event of John getting on the bus at a given point in time initiates that John is on the bus, and the event of John getting off the bus terminates that John is on the bus. Moreover, using the information expressed by the conditional statements above, a human observer can construct an explanation via abduction that tells us why John stays in Trapani. How can we model this behaviour using a machine?

3 The Event Calculus

The Event Calculus [4] and its variants [7] provide a general framework for reasoning about time and events, and can help us to solve the problems stated in the last section. The basic entities of the Event Calculus are events, fluents and time points. Events which occur at a given time point initiate fluents (= properties, states) that hold until they are terminated by other events at a later time point. I present here a version of the simplified Event Calculus [9] that uses the following two domain-independent core axioms:

```
6. holds_at(F,T2) :-
     happens(E,T1), initiates(E,F), before(T1,T2), \+ clipped(T1,F,T2).
7. clipped(T1,F,T3) :-
     happens(E,T2), terminates(E,F), before(T1,T2), before(T2,T3).
```

The first rule (6) specifies that a fluent `F` holds at a time point `T2` if an event `E` happens at a time point `T1` that occurs before `T2` and initiates the fluent `F`, and it cannot be shown that `F` is clipped (ceases to hold) between the time point `T1` and `T2`. Note that the negation as failure operator `\+` is used to specify a form of default persistence here. The second rule (7) specifies that a fluent `F` is clipped between the time point `T1` and `T3` if an event `E` happens at a time point `T2` and terminates `F`, and `T1` occurs before `T2` and `T2` before `T3`.

The relationship between events and fluents is usually modeled in the Event Calculus by a set of domain-specific `initiates` and `terminates` clauses whereas the events are represented as terms and the fluents as functions, for example:

```
 8. initiates(E,located_at(X,Y)) :- event(E,arriving(X,Y)).
 9. terminates(E,located_at(X,Y)) :- event(E,leaving(X,Y)).
```

A particular course of events is then represented as a set of ground `happens` and `event` clauses while the `before` clause keeps track of the temporal ordering:

```
10. happens(e1,‘10:10’). event(e1,arriving(sk1,sk2)). happens(e2,‘11:30’).
    event(e2,leaving(sk3,sk2)). before(‘10:10’,‘11:30’).
```

This gives us the required machinery that allows us to infer together with the formalised information of (1) that John is located at the airport of Palermo at 11:00 and, let’s say in Trapani at 14:00. In Section 5, we will slightly modify the above notation in order to integrate the Event Calculus smoothly with the formal output of the controlled language processor.

4 Speaking about Events and Effects in PENG Light

The discourse in (1) that communicates the relevant events can be expressed directly in PENG Light but this information needs to be augmented with additional domain-specific axioms that define the effects of these events for the Event Calculus. As explained above, events initiate or terminate fluents (or release them from the commonsense law of inertia [8]); we can specify these additional domain-specific axioms, for example, in the following way in PENG Light:

11. If X arrives at Y then this event initiates that X is located at Y.
12. If X gets on Y then this event initiates that X is located in Y.
13. If X is located in Y and Y leaves Z then this event terminates
    that X is located at Z.

Note that in contrast to (11) and (12), the effect in (13) is not only dependent on an event but also on a fluent that must hold in order to terminate another fluent. In PENG Light we can further restrict the domain and the range of an event by replacing the variables in the axioms by suitable nominal expressions, for example:

14. If a person gets on a vehicle then this event initiates that the person is located in the vehicle.

Note that this leads to additional constraints on the right hand side of the rules in the Event Calculus, and we need to specify additionally in controlled language that John is a person and that a bus is a vehicle.

5 Integrating PENG Light with the Event Calculus

PENG Light sentences are translated via discourse representation structures [3] into the input language language of a model builder that generates a model consisting of a number of facts. For example, the two anaphorically related sentences:

15. John arrives with Flight AZ1777 at the airport of Palermo at 10:10.
16. John gets on a bus at the airport.

result in the following model that distinguishes – among other things – between events, objects, temporal and thematic relations:

```
17. named(sk1,john). theta(e1,theme,sk1). event(e1,arriving).
    theta(e1,instrument,sk2). named(sk2,az1777). theta(e1,location,sk3).
    object(sk3,airport). associated_with(sk3,sk4). named(sk4,palermo).
    theta(e1,time,sk5). timex(sk5,‘10:10’). theta(e2,agent,sk1).
    event(e2,getting_on). theta(e2,theme,sk6). object(sk6,bus).
    theta(e2,location,sk3). theta(e2,time,sk7). timex(sk7,‘11:15’).
    before(‘10:10’,‘11:15’).
```

Two things are important to note here: first, these terms are additionally wrapped by `fact/1` (not shown for reasons of space); and second, since the sentence in (16) does not use an explicit temporal marker, a timestamp (`timex(sk7, ‘11:15’)`) is generated in our scenario during the processing of the sentence.

The following rule then sets up the interface between the facts in (17) and the domain-independent axioms (6) and (7) of the Event Calculus:

```
18. happens(E,T) :- event(E,Type), theta(E,time,X), timex(X,T).
```

The translation of the domain-specific axioms that relate the events to their effects results also in a number of rules. For example, the three domain-specific axioms (11-13) are translated into the following rules:

```
19. initiates(E,fluent(X,located_at,Y)) :-
      event(E,arriving), theta(E,theme,X), theta(E,location,Y).
20. initiates(E,fluent(X,located_in,Y)) :-
      event(E,getting_on), theta(E,agent,X), theta(E,theme,Y).
21. terminates(E,fluent(X,located_at,Z)) :-
      event(E,leaving), theta(E,agent,Y), theta(E,theme,Z),
      holds_at(fluent(X,located_in,Y),now).
```

Given this knowledge, we can now infer with the help of the Event Calculus that John is in the bus at 11:20 and still in Palermo at that time. This is because the fluents in (19) and (20) that have been initiated by the occurrence of the events in (17) continue to hold until the occurrence of further events will terminate them. For example, as soon as the information becomes available that the bus leaves the airport at 11:30, the conditions in rule (21) will trigger and terminate the fluent that John is located at the airport.

The Event Calculus can be extended in many interesting ways [7, 9]. A particular interesting extension in our context is the combination of the Event Calculus with a meta-interpreter for abductive reasoning [1, 10] in order to find explanations for *why*-questions. Abductive reasoning is inference to the best explanation; it is reasoning backwards from the consequent to the antecedent and not a valid form of reasoning, but it can suggest plausible hypotheses. Let us assume that the information in (3) is available in the knowledge base and that we know that John stays in Trapani but we would like to know why, then an abductive meta-interpreter can infer that the weather must be bad.

6 Conclusions

In this work I showed how the controlled natural language PENG Light can be extended in a systematic way to speak about events and their effects. The controlled natural language is automatically translated into the input language of a model builder. The generated formal representation can be integrated directly with the Event Calculus and be used for reasoning about events and time. This makes PENG Light an interesting specification language for creating more user-friendly interfaces to knowledge systems in dynamic domains.

References

1. Flach, P.: *Simply Logical, Intelligent Reasoning by Example.* Wiley & Sons, (1994)
2. Fuchs, N.E., Schwertel, U. and Schwitter, R.: Attempto Controlled English – Not Just Another Logic Specification Language. In: *Logic-Based Program Synthesis and Transformation, LOPSTR'98*, LNCS, vol. 1559, pp. 1–20, (1999)
3. Kamp, H. and Reyle, U.: *From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory.* Kluwer Dordrecht, (1993)
4. Kowalski, R. and Sergot, M.: A Logic-Based Calculus of Events. In: *New Generation Computing*, vol. 4, pp. 67–94, (1986)
5. Loveland, D.W., Reed, D.W. and Wilson, D.S.: SATCHMORE: SATCHMO with RElevancy. In: *Journal of Automated Reasoning*, vol. 14, no. 2, pp. 325-351, (1995)
6. Manthey, R. and Bry, F.: SATCHMO: A Theorem Prover Implemented in Prolog. In: *Proceedings of CADE 1988*, pp. 415–434, (1988)
7. Miller, R. and Shanahan, M.: Some Alternative Formulations of the Event Calculus. In: A.C. Kakas and F. Sadri, (eds.), *Computational Logic: Logic Programming and Beyond*, Part II, LNAI, vol. 2408, pp. 452–490, (2002)
8. Mueller, E.T.: *Commonsense Reasoning.* Morgan Kaufmann Publishers, (2006)
9. Sadri, F. and Kowalski, R.: Variants of the Event Calculus. In: *Proceedings of the Twelfth International Conference on Logic Programming*, pp. 67–81, (1995)
10. Shanahan, M.P.: An Abductive Event Calculus Planner. In: *The Journal of Logic Programming*, vol. 44, pp. 207–239, (2000)
11. Sutcliffe, G.: The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v.3.5.0. In: *Journal of Automated Reasoning*, vol. 43, no. 4, pp. 337–362, (2009)
12. White, C. and Schwitter R.: An Update on PENG Light. In: *Proceedings of ALTA 2009*, Sydney, Australia, pp. 80–88, (2009)