SciVerse ScienceDirect

Procedia - Social and Behavioral Sciences 27 (2011) 12 – 21

Procedia
Social and Behavioral Sciences

Pacific Association for Computational Linguistics (PACLING 2011)

Specifying Events and their Effects in Controlled Natural Language

Rolf Schwitter[a]*

[a]*Macquarie University, Balaclava Road, North Ryde, 2109, Australia*

Abstract

This paper shows how a controlled natural language can be used to construct precise formal representations for reasoning about events and their effects. Specifications written in PENG Light are translated with the help of discourse representation structures into the input language of the Simplified Event Calculus. This logic-based formalism is declarative and can be used for various reasoning tasks, among them for question answering. Taking a simple scenario from the transport domain as a starting point, we illustrate how PENG Light can be used to specify the narrative part of the scenario and the commonsense knowledge that is required to reason about direct and indirect effects of events as well as about continuous change. There is no need to encode this scenario and the background axioms in a formal notation, since the relevant information can be expressed directly on the level of the controlled natural language.

Keywords: Controlled natural language; specifications; Event Calculus; commonsense knowledge

1. Introduction

Many natural language applications would benefit from making inferences about textual information using additional commonsense knowledge that states how the world changes over time as a result of events. Commonsense knowledge comes naturally to us and helps us in everyday life to predict what happens at a given point in time and to fill in the gaps if some information is missing. Although this kind of reasoning seems to be simple for humans, it is surprisingly hard for subject matter experts without a

[a]Corresponding author.

E-mail address: rolf.Schwitter@mq.edu.au

1877-0428 © 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of PACLING Organizing Committee.
doi:10.1016/j.sbspro.2011.10.578

formal background in logic to construct precise formal representations of commonsense knowledge so that this knowledge can be used by a machine [1]. Recently, some progress has been made in learning inference rules in an unsupervised, domain-independent manner from large corpora of unrestricted text [7], but the resulting open domain theories are often too coarse-grained and do not provide the required level of detail for automated reasoning.

In order to process commonsense knowledge automatically, this knowledge is usually stated in a formal language that is expressive enough to represent commonsense entities (such as objects, events, time points and time-varying properties) and can deal with various commonsense phenomena (such as positive and negative effects of events, indirect effects, and continuous change). The Event Calculus [2] is a logic-based language for representing commonsense entities and phenomena and can be used for various reasoning tasks, among them for question answering. There exist a number of alternative formalisms that can be used for commonsense reasoning (see Mueller [2006] for an overview), but the Event Calculus is particularly interesting because of its commitment to an explicit narrative description of events. This makes it possible to derive conditions that can change over time directly from the discourse and facilitates the representation of continuous change.

Instead of using the formal language of the Event Calculus directly as a specification language, we use a controlled natural language to describe a scenario, collate the required background knowledge and bring this information into a machine-processable form. Controlled natural languages are engineered subsets of full natural languages whose grammar and vocabulary have been restricted to reduce both ambiguity and complexity of full natural languages. Recently, a number of general-purpose controlled natural languages such as Attempto Controlled English (ACE), Processable English (PENG), and Computer-Processable Language (CPL) have been developed and used as specification languages for various knowledge acquisition and representation tasks (see Schwitter [2010] for an overview). These controlled natural languages look informal at first glance, but they can often be translated unambiguously into a formal language, and their writing process is usually supported by an intelligent authoring tool [8].

The rest of this paper is structured as follows: In Section 2, we introduce a simplified version of the Event Calculus and show how this formalism can be used to specify the effects of events that occur in a scenario taken from the transport domain. In Section 3, we present the controlled natural language PENG Light and discuss how the initial scenario, the narrative part of the scenario, and the effect axioms can be specified in controlled natural language and be translated via discourse representation structures into the input language of the Simplified Event Calculus. In Section 4, we look at indirect effects of events and continuous change, and discuss how these phenomena can be described in controlled natural language. In Section 5, we summaries the advantages of our approach and conclude.

2. The Event Calculus (EC)

The Event Calculus (EC) is a narrative based temporal formalism for reasoning about actions (= events), their effects and time periods. The EC was originally introduced by Kowalski and Sergot [1986] as a logic programming framework for updating databases and for dealing with narrative discourses that describe a sequence of events. Since then many different axiomatisations have been developed for the EC, and the EC has been used successfully in the context of legal reasoning, database updates, planning, cognitive robotics, business systems, and belief representation [4, 5].

2.1. The Simplified Event Calculus (SEC)

It has been show in the literature [3, 10] that much of the representational power of the original EC can be maintained by a simpler version of the EC, known as the Simplified Event Calculus (SEC). This

simpler variant uses time points rather than time periods and can be extended to deal with continuous change; it is based on the Horn clause subset of the Predicate Calculus augmented with negation-as-failure.

The basic components of the SEC are events, fluents and time points. An event represents an action that may happen in a domain at a given point in time, such as a forklift moving a container *a* on a container *d*. A fluent is a time-varying property that depends on an event, such as the location of the container *a*. And a time point represents an instant of time on a time line, for example 07:00. After an event occurs at a given point in time the truth value of a fluent may change (e.g., the container *a* is now on the container *d*).

The language of the SEC consists of a small number of predicates (see Table 1) that allow us to specify various forms of commonsense knowledge about the effects that events may have on fluents.

Table 1. The Language of the Simplified Event Calculus

Predicate	Meaning
`holds_at(F, T)`	the fluent `F` is true at the time point `T`
`happens(E, T)`	the event `E` happens at the time point `T`
`initiates(E, F, T)`	the event `E` initiates the fluent `F` at the time point `T`
`terminates(E, F, T)`	the event `E` terminates the fluent `F` at the time point `T`
`T1 < T2`	the time point `T1` is before the time point `T2`
`initially(F)`	the fluent `F` holds from the time point zero

2.2. A Scenario from the Transport Domain

We start with a simple scenario from the transport domain to show how the SEC can be used to predict via deduction the fluents that are initiated and terminated as a result of performing a sequence of events. Initially, our scenario consists of four labeled containers (*a*, *b*, *c*, *d*) and a forklift. The initial location and state of these containers can be described as follows in Prolog:

```
1.  initially(on(c, floor)). initially(on(b, c)).
    initially(on(a, b)).     initially(on(d, floor)).
    initially(clear(a)).     initially(clear(d)).
```

Now, let us assume that the forklift first moves the container *a* on the container *d*, and then the container *b* on the container *a*. We can represent these two events and the time points when these events occur as follows:

```
2.  happens(e1, 5).  move(e1, forklift, a, d).
    happens(e2, 10). move(e2, forklift, b, a).
```

Note that the two events change the initial situation of the scenario. We use integers (`5` and `10`) here in order to represent the time points; we will switch to POSIX time notation later for describing time points. Using the language of the SEC, we can now specify the effects that these events have on fluents with the help of a number of domain-dependent axioms. These axioms are expressed in terms of positive effect predicates (`initiates/3`) and negative effect predicates (`terminates/3`). For the time being, we

represent these effect axioms for the two events that occur in our scenario in the following form (we will modify this notation later when we interface the output of the language processor with the SEC):

```
3.  initiates(E, on(X, Y), T) :-
      holds_at(clear(X), T), holds_at(clear(Y), T), X \= Y, move(E, A, X, Y).

4.  terminates(E, on(X, Z), T) :-
      holds_at(clear(X), T), holds_at(clear(Y), T), X \= Y,
      holds_at(on(X, Z), T), X \= Z, Y \= Z, move(E, A, X, Y).

5.  initiates(E, clear(Z), T) :-
      holds_at(clear(X), T), holds_at(clear(Y), T), X \= Y,
      holds_at(on(X, Z), T), X \= Z, Y \= Z, move(E, A, X, Y).

6.  terminates(E, clear(Y), T) :-
      holds_at(clear(X), T), holds_at(clear(Y), T), X \= Y, move(E, A, X, Y).
```

These domain-dependent axioms are processed in the SEC by two domain-independent **core** axioms that are implemented with the help of the following two clauses:

```
7.  holds_at(F, T2) :-
      happens(E, T1), T1 < T2, initiates(E, F, T1), \+ clipped(T1, F, T2).

8.  clipped(T1, F, T3) :-
      happens(E, T2), T1 < T2, T2 < T3, terminates(E, F, T2).
```

The first clause (7) specifies that a fluent `F` holds at the time point `T2`, if an event `E` happens at the time point `T1`, and `T1` is before `T2`, and the event `E` initiates the fluent `F` after `T1`, provided that the fluent `F` has not been clipped between `T1` and `T2`. The second clause (8) specifies that a fluent `F` is clipped between `T1` and `T3`, if an event `E` happens at `T2`, and `T2` is between `T1` and `T3`, and the event `E` terminates the fluent `F` at the time point `T2`. It is often convenient to use an additional **core** axiom to process the initial scenario (1) that consists of a number of `initially/1` predicates:

```
9.  holds at(F, T) :- initially(F), \+ clipped(0, F, T).
```

The clause (9) specifies that a fluent `F` holds at the time point `T`, if it held `initially` and has not been clipped between the time point `0` and the time point `T`. Given the domain-dependent axioms (3-6) and the core axioms (7-9), we can now investigate what the domain looks like, let's say, at the time point `T=7` using Prolog's `findall/3` predicate:

```
10.  ?- findall(P, holds_at(P, 7), L).
     L = [ on(a, d), clear(b), on(c, floor), on(b, c),
           on(d, floor), clear(a) ].
```

Other queries about the state of the domain are possible at different time points, but we will express these queries as well as the domain-dependent axioms directly on the level of the controlled natural language as we will see in the following section.

3. PENG Light

PENG Light is a controlled natural language designed for writing specification texts in an unambiguous way so that these texts can be translated into a formal target language [11]. The language of PENG Light covers a strict subset of standard English and is defined by a controlled grammar and a controlled lexicon. The controlled lexicon consists of domain-dependent content words (nouns, verbs, adjectives, and adverbs), predefined function words (e.g., conjunctions, determiners, query words), a small number of predefined fixed phrases (e.g., *there is*, *it is the case that*), and an open list of exclusion words. The author of a specification text can add new content words to the lexicon during the writing process. Simple PENG Light sentences have the following functional structure:

11. *subject + verb + [complements] + { adjuncts }*

Complements depend on the verb and are necessary constituents to complete the meaning of a sentence. Adjuncts are optional constituents that establish the circumstances under which the event or state expressed by a verb takes place. Complex sentences are built from simpler sentences through coordination, subordination, quantification, and negation. The language processor of PENG Light uses a unification-based phrase structure grammar that is processed by a chart parser [11]. The grammar contains syntactic, semantic as well as pragmatic information that is processed in parallel during the parsing process. The chart parser accepts the input and incrementally generates a discourse representation structure, resolves anaphoric references, and creates look-ahead information and a paraphrase of the input text. PENG Light only allows for a few restricted forms of anaphoric references, namely via definite noun phrases, proper nouns and variables. The resolution of these anaphoric expressions is reflected in a paraphrase generated by the language processor and displayed as feedback information for the author. The purpose of the look-ahead information is to constrain the input and to guide the writing process of the author. This look-ahead information is grouped into syntactic categories and displayed in the authoring tool of PENG Light. Whenever the author types or selects a word form that belongs to these syntactic categories, new look-ahead information is generated and displayed. This mechanism informs the author continuously about the restrictions of the controlled language (see Schwitter et al. [2003] for details). In the following, we will only use a subset of PENG Light that can be translated into Horn clause logic.

3.1. Specifying the Initial Scenario

We can specify the initial scenario in PENG Light with the help of the authoring tool that guides the writing process and accepts only well-formed PENG Light sentences:

12. *The container c is on the floor.*
13. *The container b is on the container c.*
14. *The container a is on the container b.*
15. *The container d is on the floor.*
16. *The container a is clear.*
17. *The container d is clear.*

Note that we use here a very explicit description of the initial scenario that introduces the containers with the help of a definite noun phrase followed by a name (*a*, *b*, *c*, or *d*). It is possible, although less natural in our context, to introduce these names without the preceding definite noun phrase. The

advantage of the above solution is that the noun specifies the class to which the name belongs to, and we end up with a conceptually richer specification.

3.2. Specifying Direct Effect Axioms

Apart from the initial scenario, we can also specify the relevant domain-dependent effect axioms in controlled natural language. We use conditional sentences to express the axioms (3-6) introduced in Section 2.2 in PENG Light:

18. *If a container X is clear and a container Y is clear and X is not Y and an agent moves the container X on the container Y then X will be on Y.*
19. *If a container X is clear and a container Y is clear and X is not Y and the container X is on the container Z and X is not Z and Y is not Z and an agent moves X on Y then X will no longer be on Z.*
20. *If a container X is clear and a container Y is clear and X is not Y and the container X is on a container Z and X is not Z and Y is not Z and an agent moves X on Y then Z will be clear.*
21. *If a container X is clear and a container Y is clear and X is not Y and an agent moves X on Y then Y will no longer be clear.*

Note that these conditional sentences use the continuous future tense in the consequent: the expression *will be* corresponds to the `initiates/3` predicate on the SEC level and the expression *will no longer be* to the `terminates/3` predicate.

3.3. Constructing the Narrative Text

Once the effect axioms are available, we can start to specify the narrative part of the scenario and add the relevant domain-dependent terminological information, for example:

22. *The forklift moves the container a on the container d.*
23. *The forklift moves the container b on the container a.*
24. *Every forklift is an agent.*

Note that the two event sentences (22) and (23) do not contain an explicit temporal modifier in adjunct position, but we can add one, for example: *at 07:00*, if necessary. Event sentences that do not contain an explicit temporal modifier are time-stamped during the translation into the input language of the SEC as we will see in the next section.

3.4. Translating the Scenario

The language processor of PENG Light generates discourse representation structures (DRSs) as output. The form of these DRSs is inspired by a neo-Davidsonian representation of eventualities (= events and states) and relies on thematic roles that connect events and states with other entities that occur in the discourse (see Parsons [1994] for an introduction). For example, the processing of sentence (12) results in the following DRS:

25.
```
drs([ A, B, C ],
    [ theta(A, experiencer, C), state(A, being), theta(A, location, B),
```

```
object(B, floor), named(C, c), object(C, container) ].
```

This DRS relies on a reified notation for logical atoms and on a small number of predefined predicates (e.g., `theta/3`, `state/2`, `object/2`, `named/2`). The variables `A`, `B` and `C` stand for discourse referents; `A` refers to a state, and `B` and `C` refer to objects in the domain. Thematic roles are used to connect – in our case – the state `A` with the other entities that occur in the discourse. The DRS serves as a context for the processing of the subsequent sentences and is translated into a set of facts. For example, the translation of sentence (22) first extends the existing DRS and results after Skolemisation in the following set of facts:

26.
```
event(e1, moving). theta(e1, agent, sk6). object(sk6, forklift).
theta(e1, theme, sk4). theta(e1, location, sk5).
theta(e1, time, sk7). timex(sk7, 1306047600).
```

Note that since the noun phrases *the container a* and *the container d* have already been introduced into the discourse by the sentences (14) and (15), they are now interpreted anaphorically in sentence (22). Therefore, they only occur as Skolem constants (`sk4` and `sk5`) in the above representation. Note also that the event in sentence (22) has been time-stamped using POSIX time notation `1306047600`.

Conditional sentences that contain effect predicates are translated into clauses via DRSs; for example, the translation of sentence (18) results in the following clause:

27.
```
initiates(E, [ theta(sk(X, ...), experiencer, X),
               state(sk(X, ...), being),
               theta(sk(X, ...), location, Y) ], T) :-
      theta(E, agent, A), event(E, moving), theta(E, theme, X),
      theta(E, location, Y), object(A, agent), object(X, container),
      object(Y, container), X \= Y,
      holds_at([ theta(S2, experiencer, Y), state(S2, being),
                 theta(S2, predicate, clear) ], T),
      holds_at([ theta(S1, experiencer, X), state(S1, being),
                 theta(S1, predicate, clear) ], T).
```

This example shows that the second argument of the predicate `initiates/3` and the first argument of the two predicates `holds_at/2` consist of a list of terms in contrast to the notation introduced in Section 2.2. This is a consequence of the eventuality-based notation that the language processor of PENG Light produces as output. The term `sk(X, ...)` is an abbreviation for a Skolem function that depends on the variables in the body of the clause and serves as a unique identifier for the fluent that consist of a list of terms. Note also that the order of the goals in the body of the clause has been optimised during the translation process, since we cannot assume that the subject matter expert provides the most efficient order of these goals during the specification process. A naive translation of the surface order might result in an inefficient reasoning process.

3.5. Interfacing PENG Light with SEC

The eventuality-based notation requires a few adjustments so that the domain-dependent facts and clauses can work smoothly together with the core axioms of the SEC. First, we need to add an auxiliary clause that interfaces the relevant facts derived from the narrative part of the scenario with the predicate

`happens/2` that is used in the body of the core axioms (7) and (8) of the SEC. We can do this in the following way:

```
28.  happens(E, T) :- event(E, Type), theta(E, time, S), timex(S, T).
```

A similar auxiliary clause is necessary that serves as an interface between the predicate `initially/1` used by the core axiom (9) and in the initial description of the scenario.

3.6. Answering Questions

We can ask and answer the same kind of questions about the transport domain as in Section 2.2 but now on the level of the controlled natural language, for example:

29. *What holds at 07:00?*
30. *Where is the container b?*
31. *Which container is on the container d?*

These questions are translated via discourse representation structures into the eventuality-based notation and answered with the help of the SEC.

4. Extensions

The SEC has the potential to deal with indirect effects and can be straightforwardly extended to represent continuous change as we will see in the following sections.

4.1. Specifying Indirect Effects

Consider now the case where the forklift picks up the container *b* and drives from the warehouse to the loading dock. It is obvious for a human that the container *b* will no longer be at the same location as before. That means driving from the warehouse to the loading dock has the indirect effect that the container changes its location. The problem of representing and reasoning about indirect effects of events is known as the ramification problem (see Shanahan [1997] and Mueller [2006] for an introduction).

One way to represent indirect effects is to represent them in a similar way as direct effects. We specify these indirect effects in controlled natural language and choose a suitable level of granularity for the specification. Instead of speaking about specific objects (such as forklift, warehouse, and loading dock), we want to make the specification as general as possible so that it applies to whatever we later define as an agent or as a location:

32. *If a container X is clear and the container X is on a container Y and X is not Y and an agent picks up the container X then the container Y will be clear.*
33. *If a container X is clear and the container X is on a container Y and X is not Y and an agent picks up X then X will no longer be on Y.*
34. *If an agent is at a location and an object is at that location and the agent picks up the object then the agent will be holding that object.*
35. *If an agent holds an object and the agent puts down the object then the agent will no longer be holding the object.*
36. *If an agent holds an object and the agent drives from a location X to a location Y and X is*

not Y then the agent will be at the location Y and the object will be at the location Y.

37. *If an agent holds an object and the agent drives from a location X to a location Y and X is not Y then the agent will no longer be at the location X and the object will no longer be at the location X.*

In order to use these axioms together with the initial scenario, we additionally need general inclusion axioms such as (38) and to amend the initial state descriptions (39):

38. *Every container is an object. Every warehouse is a location. Every loading dock is a location.*
39. *The forklift is in the warehouse. The container b is in the warehouse.*

Given the domain-dependent information (32-39) and the following sequence of events:

40. *The forklift picks up the container b and drives from the warehouse to the loading dock.*

we can infer that the container *b* is at the loading dock whenever the forklift is at the loading dock (holding the container) and answer question (30) successfully.

4.2. Specifying Continuous Change

It has been shown in the literature [10] that the SEC can be extended in a systematic way to represent continuous change. Let us consider the case where a truck departs at a given time point from a starting point, travels along with a fixed velocity of 80 km/h, and halts at a later time point. We can calculate how far the truck is away from the origin at each point in time, if we know the speed and the elapsed time. We can specify this form of continuous change as follows in PENG Light:

41. *If a vehicle is travelling and the vehicle is X km away from the origin at the timepoint T1 and [Y is X + (80 * (T2 - T1) / 3600)] then the vehicle will be Y km away from the origin at the timepoint T2.*

Note that the formula in square brackets is written in Prolog notation. In PENG Light formulas can occur in the conditions of conditional sentences. The translation of the axiom (41) results in a clause with a new SEC predicate on the left hand side. This new predicate (`trajectory(F1, T1, F2, T2)`) represents that if a fluent `F1` is initiated at the time point `T1`, then the fluent `F2` becomes true at `T2`.

Additionally, we need to specify the initial distance of the truck from the origin (42), the leaving event of the truck together with its time point (43), and a series of background axioms (44-48). The sentence (44) provides additional terminological information. The sentence (45) puts the vehicle into the state of travelling, and the sentence (46) terminates this state. The two sentences (47) and (48) specify what happens when a fluent undergoes the transition from a discrete fluent to a continuous one, and vice versa:

42. *The truck is 0 km away from the origin.*
43. *The truck leaves the origin at 09:15.*
44. *Every truck is a vehicle.*
45. *If a vehicle leaves an origin then the vehicle will be travelling.*
46. *If a vehicle is travelling and the vehicle halts then the vehicle will no longer be travelling.*
47. *If a vehicle is X km away from an origin and leaves the origin then the vehicle will no longer be X km away from the origin.*

48. *If a vehicle is X km away from the origin and halts then the vehicle will be X km away from the origin.*

Finally, we need to add an additional clause to the **core** axioms of the SEC that deals with the trajectory of a continuously changing quantity:

```
49.  holds_at(F2, T2) :-
       happens(E, T1), T1 < T2, initiates(E, F1, T1),
       trajectory(F1, T1, F2, T2), \+ clipped(T1, F1, T2).
```

We can translate the above sentences (41-48) and then successfully answer the following question:

50. *How many km is the truck away from the origin?*

That means the SEC can infer at any point in time how far the truck already travelled and returns the distance as answer.

5. Conclusions

In this paper we discussed the characteristics of the controlled natural language PENG Light that can be used as a high-level interface language to the SEC, a powerful formalism for reasoning about events and their effects. We started by introducing the SEC formalism and illustrating how it can be used to predict what holds at a given point in time after a sequence of events occurred. We argued that there is no need to express a scenario and the domain-dependent axioms that the SEC takes as input in a formal notation; instead we can write the specification fully in PENG Light and translate it automatically via discourse representation structures into the formal target notation. In particular, we showed how PENG Light can be used to state direct and indirect effect axioms, how to represent axioms for continuous change, and how to answer questions over the resulting formal representation. To the best of our knowledge, PENG Light is the first controlled natural language that allows an author to write a specification for the SEC in a subset of English.

References

[1] Fuchs, N. E., Schwertel, U., & Schwitter, R. (1999). Attempto Controlled English – Not Just Another Logic Specification Language. In: *Proceedings of LOPSTR'98*, LNCS, vol. 1559, pp. 1–20.

[2] Kowalski, R., & Sergot, M. (1986). Logic-Based Calculus of Events. In: *New Generation Computing*, vol. 4, pp. 67–95.

[3] Kowalski, R. (1992). Database Updates in the Event Calculus. In: *Journal of Logic Programming*, vol. 12, pp. 121–146.

[4] Miller, R., & Shanahan, M. (2002). Some Alternative Formulations of the Event Calculus. In: A. C. Kakas, F. Sadri (eds.): *Computational Logic (Kowalski Festschrift)*, LNAI 2408, pp. 452–490.

[5] Mueller, E. T. (2006). *Commonsense Reasoning*. Morgan Kaufmann Publishers.

[6] Parsons, T. (1994). *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press.

[7] Schoenmackers, S., Etzioni, O., Weld, D. S., & Davis, J. (2010). Learning First-Order Horn Clauses from Web Text. In: *Proceedings of EMNLP*, pp. 1088–1098.

[8] Schwitter, R., Ljungberg, A., & Hood, D. (2003). ECOLE - A Look-ahead Editor for a Controlled Language. In*: Proceedings of EAMT CLAW03*, pp. 141–150.

[9] Schwitter, R. (2010). Controlled Natural Language for Knowledge Representation. In: *Proc. of COLING 2010*, pp. 1113–1121.

[10] Shanahan, M. P. (1997). *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press, Cambridge, Massachusetts.

[11] White, C., & Schwitter, R. (2009). An Update on PENG Light. In: L. Pizzato and R. Schwitter (eds.), *Proceedings of ALTA 2009*, Sydney, Australia, pp. 80–88.