Modeling with Events from Policy Descriptions

Nandan Parameswaran,[1] Pani N. Chakrapani[2]
[1]School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
[2] Department of Mathematics and Computer Science, University of Redlands, Redlands, CA 92373, USA

Abstract

Given the policy descriptions in the domain of vehicle insurance, we have attempted to provide an event based model of the descriptions using event diagrams and an event description notation. Event models show events and time explicitly thus enabling intuitive reasoning easier for non-experts. We quantify the overhead in reasoning with events by measuring the mental effort.

Introduction

Policy is a set of principles that guide agents in making decisions to perform socially acceptable behaviors. In several domains, policies are described in natural languages such as English. Event based models of policy descriptions make the concepts and constructs easy to understand as these models clearly tell what events happen and when, what the states of the resources are, and the history and commitment in future an agent has. In this paper, we have made an attempt to provide an event based model for concepts occurring in the domain of vehicle insurance policy written in informal English.

In the following sections we first define events, the relations that exist between them, and a simple (pseudo) notation to represent complex event structures. We then consider the domain of vehicle insurance policy descriptions, identify several types of objects, events, and relations and use them to provide event models of the core concepts in the domain. We also estimate the overhead in reasoning with events using the concept of mental work. In the last section, we compare our work with other related work, and conclude with final remarks.

Events

An event ei is said to occur when an object changes its state from sj to sk, denoted as, ei::sj→ sk. As events involving an object occur in time, a timeline represents a sequence of events that the object is subjected to, and thus each object has a timeline of its own. We group events and states occurring in one or more timelines (involving several objects) and call the group as an event structure. An action is an event structure when it is intended by an agent as an action. A timeline is divided into two parts: the past and the future, and there is a unique point NOW that represents the current time instant. Finally, there is one global timeline on which each timeline is projected to for reasoning purposes when necessary. Figure 1 shows an event diagram where events

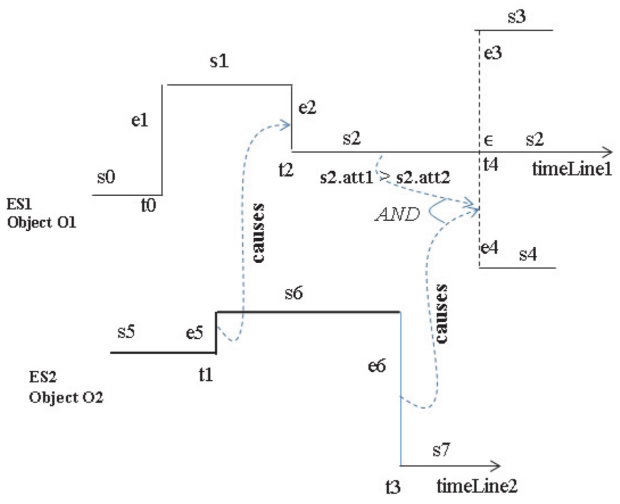

Figure 1: An event structure diagram

occurring in two objects O1 and O2 over their individual timelines are shown. In object O1, event e1 changes the state s0 to s1 at the time instant t0. At t4, the object O1 has three options: e3, ϵ, and e4 where ϵ refers to null event. Any one of the three events can happen, but the occurrence of ϵ does not change the state of the object. The occurrence of an event may be related to states that currently exist or events that occurred in the past (as depicted by the arcs in the diagram). The occurrence of e4 is related to s2 and e6: *(((s2.att1 > s2.att3) preCondition) and (e6* causes)) e4. The expression s2.att1 refers to the attribute value of an attribute att1 of s2. When the precondition is true, e4 is caused by e6. A context for an event ei occurring at time instant tj is a set of states that exist at tj where ei is related to each state sk by some relation Rp. For ei to occur at tj, the condition (sk Rp ei) is checked at tj for each sk by accessing the attribute values of sk currently available. However, when ei is related to an event ej that occurred at tj in the past, the agent has to remember the occurrence of ej (in its event memory) before performing ei leading to performance overhead.

Figure 2 shows the description of events in Figure 1 using a simple event structure notation. (We use this notation often instead of diagrams to conserve space.) ES1 is the event structure describing the events for object O1 where t0, t2 and t4 are time instants, s0,s1,s2,s3,and s4 are states. Events occur in the sequence specified: for example, e1 at t0, and e2 at t2. The *where* clause specifies the relation between events and states.

TimeConstraints t0 $<$ t1 $<$ t2 $<$t3 $<$ t4
EventStructure ES1 /* on timeLine1 */
Object O1
Sequence
t0: e1:: s0 $\rightarrow$ s1
t2: e2 :: s1 $\rightarrow$ s2
where ES2.e5 causes e2
/* ES2.e5 denotes event e5 from ES2.*/
t4: e3:: s2 $\rightarrow$ s3 | e4:: s2 $\rightarrow$ s4 | ϵ :: s2 $\rightarrow$ s2
where ((ES2.e6 causes and ((s2.att1 $>$ s2.att2) preCondition)) e4

EventStructure ES2 /* on timeline2 */
Object O2
Sequence
t1: e5 :: s5 $\rightarrow$ s6
t3: e6 :: s6 $\rightarrow$ s7

Figure 2: Event structure notation for the events in Figure 1.

Thus, ES2.e5 *causes* e2 denotes the fact that e2 is causally related to e5 from the event structure ES2. The vertical bar | stands for option. At time instant t4, three events e3, e4 and ϵ are possible but only one event is permitted to happen. The occurrence of e4 is subject to the condition specified being true at t4. Finally, the constraints across time instants in different event structures are specified by the relations in the declaration **TimeConstraints**.

Modeling policy descriptions with Events

We propose the following steps for event based modeling for a given vehicle insurance policy description.

Step 1: Identify objects and agents

The motor insurance policies refer to several objects categorized into different types. Broadly there are two types of objects: world objects such as vehicle, policy document, etc., and a mental object. Often the world objects are organized hierarchically. Some objects can be viewed as basic and others as composite composed of simpler objects. Each object has a set of attributes. Further, objects are dynamically created and destroyed at the end of its lifetime. Objects have relations amongst themselves, such as *has, part-of* and *adjacent-to*, and with agents such as *insured-by* and *driven-by*. In addition, the policies also refer to a few agents of which we only consider two: the insurer and the insured.

Step 2: Identify events

We identify two types of events: world events and mental events. World events involve world (physical) objects and mental events change the internal states of agents affecting their present and future behaviors. Events can be simple or composite. A simple event consists of one basic event and a composite event may consist of several events involving one or more objects and agents. While we have viewed the world as consisting of several named objects participating in event occurrences, we have used anonymous mental objects while referring to mental events.

Step 3: Identify states

When the values of the attributes of an object are affected by an event, state changes occur. Information of the past is partially stored in each state, but when a state is terminated, this information may be lost. In such cases, the agent has to remember the past in its event memory. Ideally, an agent should remember the entire past and this requires that all events that occur to the objects be remembered. However, in practice the agent only has to remember those events in the past that are used in reasoning rules. Similarly, all events that are committed to and in future used in reasoning rules must also be remembered.

Step 4: Identify relations

Events and states of an object are related within themselves and with events and states of other objects. The presence of relations make reasoning over large intervals of time involving several events difficult.

Step 5: Abstracting events and defining event structures

A sequence of events and states can be abstracted either to form an abstract event or an abstract state. Similarly, we can define events and states of a composite object in terms of the events and states of the component objects for reasoning purposes. Thus, timelines of component objects can be merged to produce the timeline of a composite object. Applying this technique iteratively, we can build complex event structures.

In practice an event structure is formed using sequences of events and states from one or more timelines over a carefully chosen interval of time. Each event structure can be viewed as an event module that helps simplify reasoning, and they have the traditional advantages of modules in an engineering design. Modules along with parameterized abstractions help in recursively building larger modules. Viewing timelines in terms of modules makes it easy to understand and manipulate with them. Modules can occur parallelly or sequentially. In parallel compositions of modules, events and states of one module may depend on the events and states of the other at any time instant. In a sequential composition, the preceding module, sets up states (through which data flows) for all the succeeding modules to use. It may also set up events that may be recurring in future. To simplify reasoning over large intervals of timeline, modularity may be improved by minimizing the relations defined across modules.

Step 6: Design rules

In this final step we use objects, events, states and event structures to design rules. A rule is of the them LHS → RHS, where LHS is a state based condition, event based condition, or combination of both, and RHS is an event or an event structure. We model the policy descriptions using a set of rules and initial conditions involving events and states. Rules are also useful in defining domain constraints.

Event Trace

An event trace for an object is a sequence of events that has occurred along its timeline from the beginning to NOW. An event trace is produced: (i) by applying a set of rules consecutively on the given initial state of the object s; and (ii) by the environment in which the object is situated. Whenever the antecedent of a rule matches with events and states on the timeline, the consequent of the rule is used to construct the next sequence of events and states on the timeline. In practice, it is useful to show explicitly on the event trace all future events that an agent is committed to and all options that were examined at each time instant in the past.

The NRMA policy descriptions

The motor insurance policy descriptions we specifically consider in this paper was published by The National Roads and Motorists Association(NRM). The central themes in this policy description involve events and time. Major events involve buying policy, accidents, cover, making claims and exclusions. Policy period is twelve months, and all entities (including policy, objects,commitments, and agents) have a maximum lifetime of one year. Events such as reporting an incident, making claim, paying premiums, etc. have deadlines, and follow patterns with pre-specified durations. At many points along the timelines, there are options available for agents to choose from. Agents have responsibilities and rights about what to expect in future on their timelines. Objects have composite attributes, composed of simpler objects, and they are created and destroyed dynamically. Limited common understanding between agents is achieved by demanding clarifications and simple negotiations. Time is continuous for each timeline and is present explicitly and pre-structured as months but there are also structures induced by events. We have identified the following items categorized as objects, attributes, etc. as shown in Table 1.

Organization of the policy document

The overall organization of the insurance policy description consists of six major parts: (i) agreement, (ii) insurance details (including loss or damage to vehicles, benefits, optional cover, liability cover, and additional feature), (iii) exclusions, (iv) responsibilities, (v) claims, and (vi) information about policy cancellation, complaints and resolving disputes. Each part uses a set of composite objects, attributes, event structures, mental events and mental states to describe the policy. Three agents explicitly addressed in the policy descriptions are: the policy issuing agent (often referred to as We, the insurer), the policy buying agent (referred to as You, the insured), and the driver of the vehicle. (For simplicity, we have assumed that the insured and and the driver are the same.) There are two major event modules where events in them span over one year: the **Agree Event** module that happens first, and the **Cover Event** module that follows. The Agree Event module is important because it is in this module that the We agent and the You agent enter into mental states of committing to a sequence of future events. The Cover Event module is composed of events that occur in response to an accident in the world. We thus initially have the following ES structure as shown in Figure 3.

Table 1: *Primary items from the NRMA policy descriptions*

Modeling construct	**Mental item**	**World item**
Objects	policy,insurance, contract	vehicle
Attributes	agreed value, excess,limits, premium,total-loss,responsibility	
States	cover, accidental damage	damage, accidental damage, property- damage
Events	insure, cover,agree,pay-premium,cancel	pay premium
Temporal objects	month,year, deadline, cooling-off-period	
Event structures(ES)	benefits, responsibility, complaints	benefits
Event trace (ET)		incident
ES rules	claim,exclusions, liability cover	claim,exclusions, liability cover
Constraints	limits	
Time	policy period, deadline	
Agent	insured/driver, insurer	insured/driver, insurer

```
TimeConstraints t0 < t1
EventStructure Insurance()
Sequence /* agree and cover are mental events */
t0: agree :: noAgreement → haveCover
t1: cover :: haveCover → coverProvided
```

Figure 3: Overall structure of the policy document.

The events *agree* and *cover* are abstract collective mental events that occur at abstract time instants t0 and t1. When the events are refined using lower level events, time instants t0 and t1 also get refined spreading over several refined time instants. The state *haveCover* is an abstract state and some of the substates of this state continue to provide the necessary context and conditions for the future events (even after

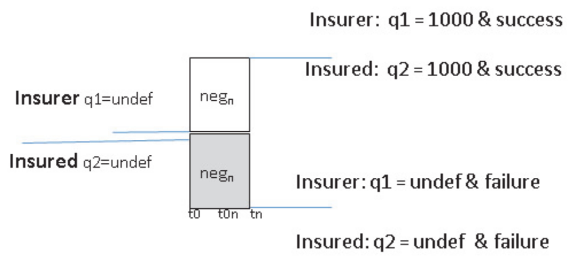
Insurer: q1 = 1000 & success
Insured: q2 = 1000 & success
Insurer q1=undef
Insured q2=undef
neg$_n$
neg$_n$
Insurer: q1 = undef & failure
t0 t0n tn
Insured: q2 = undef & failure

Figure 4: Abstract event $negotiate_n$

haveCover has been terminated) to produce the state *coverProvided*. In this event structure, the We agent and the You agent collectively respond as one agent to the events.

The Agree event module

The events in this module relate to: limits, exclusions (what are not covered), agreed values, premium amount and payment, what happens if premium is not paid, and the rights and responsibilities of the We agent and the You agent. Specific to them are mental events and mental states that forbid certain world events and world states. In particular, they relate to incidents that occur outside the policy period, driver and vehicle violating pre-specified conditions and intentional acts. Further an object called *contract* is created, maintained and finally destroyed(after a year).

Representative statements in Agree Event module

We will now illustrate how modeling is done for a few selected important statements from the descriptions.

Statement
The agreed value is the amount we agree to insure your vehicle for.

In this example, we illustrate the use of abstract events in modeling complex processes such as negotiation using sequences of mental states. In the event diagram in Figure 4 we show *negotiate$_n$* as an abstract event (which can be refined in terms of lower level events and states later) occurring at an abstract time instant t0n, the occurrence of which changes the mental states of both the agents.

To start with, both the agents begin with their mental states in which the initial agreed value given by the attribute *quote* is undefined. After n iterations of negotiations, the agents either mutually agree on a final value or reject the quote and let it remain undefined. The insurer accepts if quote $>$ 1000, and the insured accepts if the quote $\leq$ 1000. It may be noted that the mutual agreement here is formed in terms of each agents local mental states and is an approximation of the traditional mutual belief based agreement(Levesque, Cohen, and Nunes 1990). The event description in our notation is shown in Figure 5.

The abstract event negotiate$_n$ is shown as a shared event that is performed by both the agents. However, when the event is refined, each agent performs its own private events. In Figure 6 we refine negotiate$_n$ in terms of private events

TimeConstraint t0 $<$ tn; t0n = $<$ t0, tn$>$
/* $<$ to, tn $>$ denotes interval */
EventStructure ESo (n)
Agent Insurer
t0n: neg$_n$:: q1 = undef $\rightarrow$ q1 = 1000 and success
  | neg$_n$:: q1 = undef $\rightarrow$ q1 = undef and failure
EventStructure ES1(n)
Agent Insured
t0n: neg$_n$:: q2 = undef $\rightarrow$ q2 = 1000 and success
  | neg$_n$:: q2 = undef $\rightarrow$ q2 = undef and failure

Figure 5: Abstract action negotiate$_n$.

quote, agree and donotAgree and shared event neg$_{n-1}$. It is a simple negotiation where the We agent (insurer) quotes a figure at t0, the You agent (insured) agrees or does not agree, the insurer modifies it, and so on, until they arrive at a figure that each one agrees upon. It may be noted that during the recursive event occurrences, new time instants labeled as t1 and t23 are created each time.

TimeConstraint t0 $<$ t1 $<$ t2 $<$ t3 $<$ tn; t0n = $<$ t0, tn$>$
  /* t0n denotes time interval */
EventStructure ES0 (n)
Agent Insurer
t0: quote:: q1 = undef $\rightarrow$ q1 = v1
  where ES1.q2 = undef preCondition quote
t2: finalize:: q1 = v1 $\rightarrow$ q1 = v1 and success
  | rejectANDquote :: q1 = v1 $\rightarrow$ 4 q1 = v3
  where ES1.q2 = v1 preCondition finalize and
  ES1.q2 = c preCondition rejectANDquote
t3n: neg$_{n-1}$ :: q1 = v3 $\rightarrow$ q1 = vn and success
  | neg$_{n-1}$ :: q1 = v3 $\rightarrow$ q1 = undef and failure

EventStructure ES1 (n)
Agent insured
t1: accept :: q2 = undef $\rightarrow$ q2 = v1 and success
  | rejectANDquote :: q2 = undef $\rightarrow$ q2 = v1
  where ES0.q1 $<=$ preCondition accept and
  ES0.q1 $>$ c preCondition rejectANDquote
t3n: neg$_{n-1}$ :: q2 = c $\rightarrow$ q2 = vn and success
  | neg $_{n-1}$ :: q2 = c $\rightarrow$ q2 = undef and failure

Figure 6: Negotiate for a value.

The agreed value is available in the mental state of the participating agents as quote = vn.

The Cover event module

The next major event module in the policy descriptions is the Cover Event module. Events in this module occur primarily in response to incidents in the real world (such as an accident) and include reporting and making claims as permitted by the allowable benefits. This module also contains events related to loss or damage to vehicles, benefits guaranteed by the We agent, exclusions and responsibilities, claims, cancelling policy (by the You agent or We agent), and resolving complaints and disputes that occur. These events occur in the context of ongoing events set up by the previous Agree Event module. When the insurance policy is cancelled, the context (consisting of mental events and states) is terminated leading to the termination of events and states in the cover module.

Representative statements in Cover Event module

One important concept that keeps occurring in this domain is *what if something does not happen*. Another important concept is exclusions which refer to situations when cover is not provided. These are situations where negative events occur, such as *we dont*, *it will not*, etc. Negative events are those that prevent certain future (positive) events and states from happening.

Statement
We do not provide cover if the driver of your vehicle was under the influence of any alcohol or drug.

The occurrence of a negative event results in turning on the mental state where the agent is committed to not selecting an event that favors certain positive states in future. While this is hard to do in general, fortunately in this domain it is fairly straightforward if we use event modeling. In Figure 7, the *donotCover* event makes the agent to commit to not-cover in future which results in sending a regret message when an incident occurs where the driver was found drunk. The event model also implements a protective mechanism where in a situation when neither *do not cover* nor *cover* happens, the current initial state becomes unstable leading to an error state at t1 thus preventing any inadvertently offered *cover* in future.

TimeConstraint t0 $<$ t1 $<$ t2
EventStructure ES0
t0: doNotCover :: initialState $\rightarrow$ commitToNotCover | cover :: initialState $\rightarrow$ coveredState
| ϵ :: initialState $\rightarrow$ initialState
t1: someEvent :: initialState $\rightarrow$ errorState
EventStructure ES1
t2: incident :: driving $\rightarrow$ incidentHappened

EventStructure ES2
t3: sendRegret :: waitState$\rightarrow$ regretSent
| respond :: waitState $\rightarrow$ serviceState
where (incidentHappened *driverDrunk* and commitToNotCover precondition) sendRegret

Figure 7: Exclusions — influence of alcohol or drug.

Finally, in this domain, the concept of responsibility is emphasized strongly. The event that occurs in this context is *must*. We can model this using a mental event which sets the agent's mental state to *committed-to*.

Mental Effort and Mental work

Since events occur over time, reasoning about them need explicit references to events that occurred in the past and to future events that the agent is committed to. Thus past and future events must be maintained in the event memory and this results in computational overhead. We measure this overhead using two quantities: mental effort and mental work(Parameswaran and Chakrapani 2014). Metal effort, denoted by eff_m at any time instant is measured by a quantity proportional to the number of events that are currently held in the event memory.If the mental effort eff_m occurs over a time period T, then we say that the mental work done by the agent over T is proportional to eff_m * T.

Mental effort due to a single event

Let e1 be an event that occurred at time t1. The mental effort at time instant NOW due to this event is m where m is a constant. The mental work spent on this occurrence up to NOW is work_m (e1, t1, NOW) = k* m* (NOW - t1) for some constant k.

Mental effort due to multiple events

For a sequence of two events $<$e1;e2$>$ where e1 occurred at t1 and e2 occurred at t2, the total mental effort at NOW is given as eff_m($<$e1;e2$>$,NOW)= eff_m(e1,NOW) + eff_m(e2,NOW) = 2m. Thus, eff_m($<$e1;...;en$>$) = $\sum_{i=1}^{n} eff_\text{m}(ei) = m * n$. Then, work_m($<$e1;...;en$>$) can be shown to be O(n^2).

Mental effort in recurring events

When recurring events occur, say n times, the overhead due to these events over the duration of n units of time is the mental effort proportional to O(n^2). However, if events are a nested r times, then the mental effort will be O(n^{2*r}).

Simulation

We consider an agent that has to reason about its past and entire future before performing any current event. Given a timeline of finite length, as the time progresses, the size of the history increases and the future decreases. However, the agent has to reason about all the options that it has in its future.

Figure 8 shows the mental effort(E) and mental work(W) spent in different scenarios (simulated in NETLOGO 2016) over time. The labels 1, 20 and 40 indicate the number of options open at any time instant along the timeline. As the options increase, both the effort and work increase. As the future option decreases to 1, the effort remains constant and the mental work increases.

Related Work

Modeling English statements using events has long been pursued by linguists, philosophers and computer scientists. Davidson(Davidson 1967) observed that simple event sentences can be analysed by existentially quantifying the action variable in the sentences. Parsons used events to understand natural language sentences(Parsons 1990). While we have used event diagrams and event notation to describe complex event structures, event calculus is also a tool to

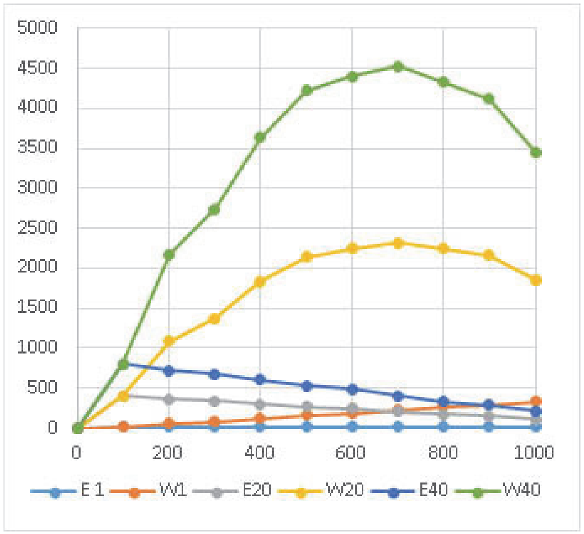

Figure 8: Mental effort (E) and mental work (W). Numbers show the number of options available at any time instant

represent events and reason about them in the logic domain(Mueller 2014). Controlled natural languages (CNL) is a carefully chosen subset of natural language that is used for policy descriptions(Schwitter 2010). There have been no attempts so far to produce event based model of policy descriptions from CNL. In (Abrahams, Eyers, and Bacon 2004) event modeling for e-commerce applications has been reported. Ponder is an object-oriented language that was proposed for specifying security management policy for distributed systems(Damianou et al. 2001). Event based semantics within the framework of a logic based formalism is reported in(Tan and Thoen 2002). Event extraction from natural language texts has gained attention recently(Hogenboom et al. 2011) where events that mostly correspond to verbs are extracted using machine learning techniques. In (Cybulska and Vossen 2011) for example, events are extracted from historical archives, by modeling events using four slots $<$ location, time, participant,action $>$. More recently, Zarri has proposed a method to ”isolate and represent” events in a stream of narrative information(Zarri 2015) using a knowledge representation language called NKRL (the Narrative Knowledge Representation Language). We have used a pictorial representation and a procedural notation in our modeling process. Further we have shown explicitly the relationship that exists between mental events and the world events as they happen over time.

Discussion

The advantages of event based model are in its explicit representation of events, states and time thus making it easier for an application designer and the insured agent to understand and reason with. We have presented our model using event diagrams and a simple event based notation which was more expressive in representing abstractions. We have not addressed the role of English prepositions in our model. Further, more attention needs to be paid to relations that are embedded in the policy description.

References

Abrahams, A. S.; Eyers, D. M.; and Bacon, J. M. 2004. An event-based paradigm for e-commerce application specification and execution. In *Proceedings of the 7th International Conference on Electronic Commerce Research (ICECR7)*, 181–192.

Cybulska, A., and Vossen, P. 2011. Historical event extraction from text. In *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, 39–43. Association for Computational Linguistics.

Damianou, N.; Dulay, N.; Lupu, E.; and Sloman, M. 2001. The ponder policy specification language. In *Policies for Distributed Systems and Networks*. Springer. 18–38.

Davidson, D. 1967. The logical form of action sentences.

Hogenboom, F.; Frasincar, F.; Kaymak, U.; and De Jong, F. 2011. An overview of event extraction from text. In *Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011) at Tenth International Semantic Web Conference (ISWC 2011)*, volume 779, 48–57. Citeseer.

Levesque, H. J.; Cohen, P. R.; and Nunes, J. H. 1990. On acting together. In *AAAI*, volume 90, 94–99.

Mueller, E. T. 2014. *Commonsense reasoning: an event calculus based approach*. Morgan Kaufmann.

Comprehensive Car Insurance Policy. https://www.nrma.com.au/car-insurance/comprehensive-car-insurance. Accessed: 2016-08-25.

Parameswaran, N., and Chakrapani, P. N. 2014. User intentions in interactive tasks. In *International Conference on Circuits and Systems (ICCS'14)*.

Parsons, T. 1990. Events in the semantics of english: A study in subatomic semantics.

Schwitter, R. 2010. Controlled natural languages for knowledge representation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 1113–1121. Association for Computational Linguistics.

Tan, Y.-H., and Thoen, W. 2002. Using event semantics for modeling contracts. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, 2198–2206. IEEE.

Zarri, G. P. 2015. A structured and in–depth representation of the semantic content of elementary and complex events. *International Journal of Metadata, Semantics and Ontologies* 10(1):12–27.