A Case Study in Reasoning about Processes

CHUNPING LI
School of Software, Tsinghua University
Beijing 100084
CHINA
cli@tsinghua.edu.cn

Abstract - In this paper, we give a case study to show how a high level semantics of processes can be integrated with the event calculus to reason about complex, continuous processes. We present a formal method to specify the semantics of processes in the event calculus and implement the automated reasoning about processes and continuous change in the logical programming framework.

Key-words: Reasoning, Planning, Event Calculus, Processes, Semantics, Logic Programming

1. Introduction

It is known that logical reasoning aims at solving problems, or proving validity. Calculi provide a way of testing the validity of formulas in a purely mechanical way [5]. In the aspect of reasoning about dynamic, continuous systems, the research standpoint concentrated on specialized logical formalisms, typically of the event calculus and its extensions [15, 16, 9, 18, 20, 3, 12, 22].

The event calculus is a logical formalism presented by Kowalski [10] for reasoning about time and events. It uses general rules to derive that a new property holds as the result of the event and associates local time periods. In [16, 20, 21, 3], similar attempts based on the logical formalisms of the event calculus have been exploited for representing continuous change. However, these ideas have not yet been exploited to define a high level action semantics serving as basis for a formal justification of such calculi, their comparison, and an assessment of the range of their applicability.

Whereas these previously described formalisms have directly focused on creating new or extending already existing specialized logical formalisms, the other research consists in the development of an appropriate semantic [7, 19, 24] as the basis for a general theory of action and change, and successfully applied to concrete calculi [2, 6, 23]. In [7], the *Action Description Language* was developed which is based on the concept of single-step actions, and does not include the notion of time. In [19], the duration of actions is not fixed, but an equidistant discretization of time is assumed and state transitions only occur when actions are executed. In [24], it is allowed for user-independent events to cause state transitions. Again equidistant discretization is assumed. But these formalisms are not suitable for calculi dealing with continuous change.

This paper builds on the work of Herrmann and Thielscher [8] and Shanahan [20, 22] to integrate a high-level semantics of processes with the event calculus to reason about continuous change. Our aim is to overcome some of limitations of the earlier works and furthermore, to present the possibility of embedding the high level semantics of processes in the logical programming framework. With a case study we show how the high level semantics of processes can be integrated with the event calculus to reason about complex processes and continuous change. In section 2, an example domain is introduced, and a conventional mathematical model is constructed in some detail. Section 3 will describe how to represent the semantics of processes in the event calculus. In section 4, logical programming is implemented to support the semantics of processes with the event calculus. In section 5, we have a summary for this work.

2. A Case Study

We illustrate how an example, the interaction between a pendulum and balls that travels along a

1-dimension space, can be formalized. As described in Fig.1, a pendulum collides at angle $\varphi = 0$ with a ball being at position $y = y_c$ at the same time. We need to find appropriate equations describing various possible movements and interactions.
Supposing the damping factor is neglected, the motion of the pendulum can be described by the following differential equation.

$$m \cdot l^2 \cdot \frac{d^2\varphi}{dt^2} = -m \cdot g \cdot l \cdot \sin\varphi - l^2 \cdot \frac{d\varphi}{dt}$$

where *l* is the length of the pendulum, *m* is the mass of the pendulum, and *g* is $9.8 m/s^2$. Solving the differential equation results in the angle of the pendulum φ, the angular velocity φ' and the angular acceleration φ''.

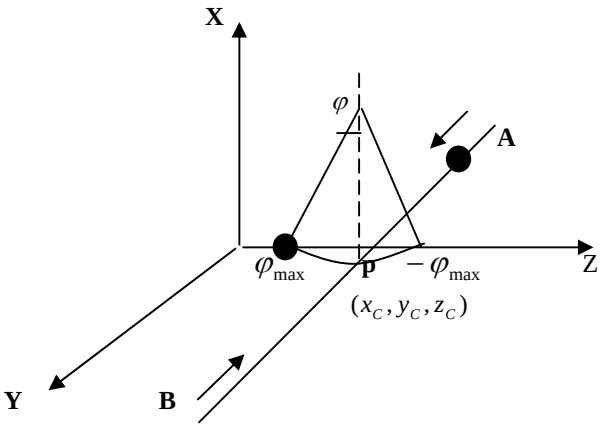

Fig. 1. Pendulum P and balls A and B in positions

As the process scheme for the motion of the pendulum we obtain $\tau_{PendulumP} = \langle C', F' \rangle$ where $C' = \{\varphi_{\max}, \gamma, y_c\}$ and $F' = \{\varphi, \varphi', \varphi''\}$.
Here we define two different types of events. The first is the collision of two balls *A* and *B*, caused by identical locations at a certain time. The second type of event is the collision between one of the balls and the pendulum *P*, defined by the angle of the pendulum being zero while the ball's position is at the y-axis position of the pendulum y_c, at the same time.
The pendulum is assumed to be of much larger mass than the balls, such that the collision will simply be an elastic impact with one of the balls (reflection into opposite direction) while the pendulum keeps moving continuously.

3. Axiomatization

3.1 Event Calculus with Circumscription

The event calculus [10] was developed as a theory for reasoning about time and events in a logical programming framework. It focuses on the concept of an event as highlighted in a semantic network representation of case semantics [11]. In the event calculus, the ontological primitives are *events* and *properties*. Properties are initiated and terminated by events. The property, which has been initiated, continues to hold by default until some event occurs which terminates it. Therefore, it is concerned with formalizing the effect of events on objects, and their properties.
In general, the fact that a property holds for a period of time can be derived from an event description by means of the initiation and termination rules:

Holds(*after*(*e, p*)) ← *Happens*(*e, t*), *Initiates*(*e, p*).
Holds(*before*(*e, p*)) ← *Happens*(*e, t*) , *Terminates*(*e, p*).

where the occurrence of an event *e* at time *t* is denoted by *Happens*(*e, t*). The formula *Initiates*(*e, p*) (or *Terminates*(*e, p*)) means that event *e* initiates (or terminates) the property *p*.
A property holds at a given time point if it holds for a period including that time point. This can be formalized by employing a predicate *HoldsAt*(*e, n*). We can define it by way of the following expressions.

HoldsAt(*p, t*) ← *Holds*(*after*(*e, p*)), *In*(*t, after*(*e ,p*)).
HoldsAt(*p, t*) ← *Holds*(*before*(*e, p*)), *In*(*t, after*(*e ,p*)).
In(*t, r*) ← *Start*(*r, e*), *End*(*r, e'*), *Time*(e', t_1), *Time*(e', t_2), $t_1 < t < t_2$

Shanahan presented a full predicate calculus version of event using circumscription in [20]. In his enriched version of event calculus, a many-sorted language of the first-order predicate calculus with equality is used, including variables not only for time $(t_1, t_2, ..., t_n)$, events $(e_1, e_2, ..., e_n)$ and properties $(p_1, p_2, ..., p_n)$ but also states $(s_1, s_2, ..., s_n)$ and truth elements $(f_1, f_2, ..., f_n)$. The domain of truth-values has two members, denoted by the constants *True* and *False*. A pair $\langle p, v \rangle$ is a truth element. A state is represented as a set of truth elements. The property *p* in the given state *s* are

described by the predicate *HoldsIn* with the following axioms.

$HoldsIn(p,s) \leftarrow [\langle p, True\rangle \in s \wedge \neg Abstate(s)]$. (E1)

$HoldsIn(p,s) \leftarrow [\langle p, False\rangle \in s \wedge \neg Abstate(s)]$. (E2)

The following axiom defines the predicate *State*. The formula *State*(*t, s*) represents that time point *t* is associated with the state *s*. Each time point is associated with a single, characterizing state *s*, such that $\langle p, True\rangle \in s$ if and only if the property *p* was initiated by some event before *t* and still holds at *t*, and $\langle p, False\rangle \in s$ if and only if *p* was terminated by some event before *t* and still does not hold at *t*.

$State(t, s) \leftrightarrow (\forall p)[[\langle p, True\rangle \in s \leftrightarrow Initiates(p, t)] \wedge [\langle p, False\rangle \in s \leftrightarrow Terminates(p, t)]]$. (E3)

3.2 Representing the Process Semantics of in the Event Calculus

The process description of the balls is formalized into rules as follows.

$HoldsAt(moving(Ball, x, (l,v,t)), t) \leftarrow$ (PS1)
$Holds(after(e, engine(Ball, F, x, (l,v,t)))), time(e, t_0)$,
$In(t, after(e, engine(Ball, F, x, (l,v,t)))), t_0 \le t$,
$State(t, s), HoldsIn(engine(Ball, F, x, (l,v,t)))), s)$,
$ContinuousProperty(engine(Ball, F, x, (l,v,t)), t_0,$
$moving(Ball, F, x, (l,v,t)), t)$.

$ContinuousProperty(engine(N, F, x, (l, v, t_0)), t_0,$ (PS2)
$moving(N, x, (l, v, t_0)), t) \leftarrow x = l + v \times (t - t_0)$.

The process description of the pendulum is formalized into rules as follows.

$HoldsAt(angle(PendulumP, \varphi, (\varphi_{\max}, \gamma, y_c)), t) \leftarrow$ (PS3)
$Holds(after(e, swing(PendulumP, F', \varphi,$
$(\varphi_{\max}, \gamma, y_c)))), time(e, t_{P0})$,
$In(t, after(e, swing(PendulumP, F', \varphi,$
$(\varphi_{\max}, \gamma, y_c)))), t_{P0} \le t$,
$State(t, s), HoldsIn(swing(PendulumP, F',$
$\varphi, (\varphi_{\max}, \gamma, y_c)), s)$,
$ContinuousProperty1(swing(PendulumP, F',$
$\varphi, (\varphi_{\max}, \gamma, y_c)), t_{P0},$
$angle(PendulumP, \varphi, (\varphi_{\max}, \gamma, y_c)), t)$.

$ContinuousProperty1(swing(PendulumP, F',$
$\varphi, (\varphi_{\max}, \gamma, y_c)), t_{P0}, angle(PendulumP,$
$\varphi, (\varphi_{\max}, \gamma, y_c)), t) \leftarrow$ (PS4)
$\varphi\ \varphi = -\varphi_{\max} \cdot \cos(\frac{2\pi}{\gamma} \cdot (t - T_{P0}))$.

where T_{P0} denotes the starting time of the motion of the pendulum.

The event of the collision between the pendulum and any of the balls *A* and *B* can be transformed into the rules:

$ImplicitHappens(e,t) \leftarrow$ (PS5)
$Start(after(e, engine(Ball, F, x, (l_{new}, v_{new}, t)))), e)$,
$End(after(e', engine(Ball, F, x, (l_{old}, v_{old}, t_{old})))), e), e' < e$,
$Holds(after(e, swing(PendulumP, F',$
$\varphi, (\varphi_{\max}, \gamma, y_c))))$,
$ConstraintRelation(l_{new}, v_{new}, l_{old}, v_{old}, \varphi_{\max}, y_c, t_0, t_{P0}, t)$.

$ConstraintRelation(l_{new}, v_{new}, l_{old}, v_{old}, \varphi_{\max}, y_c, t_0, t_{P0}, t) \leftarrow$
$l_{new} = y_c, v_{new} = -v_{old}, v_{old} \neq 0,$
$t = (y_c - l_{old}) / v_{old} + t_0$. (PS6)

where t_0 denotes the initial time of moving balls *A* or *B*.

We suppose that ball *A* starts from position 0 at time 2*sec* to move with speed 0.4*m/sec*, while ball *B* starts from position 4*m* at time 4*sec* with speed -0.3*m/sec*. If there is no other event to occur, the two balls *A* and *B* which move toward each other along the y-axis would have a collision at time 10*sec*.
Suppose that we start the pendulum with suspension point x_c = 1*m*, y_c = 0.3*m*, z_c = 0, that for the time constant we have γ = 1 and for the starting angle $\varphi = 10^\circ$ at time 1*sec*. *MoveB* and *MoveP* denote to the events which cause the balls and the pendulum to move, respectively. We have the following domain-dependent formulae.

$Happens(MoveP, 1sec)$. (PH1)
$Happens(MoveA, 2sec)$. (PH2)
$Happens(MoveB, 4sec)$. (PH3)

Let χ be the conjunction of the axioms *PS*, *PH*, and *E* without *E3*. $CIRC_{ec}[\chi]$ yields

$Abstate(s) \leftrightarrow [\langle p, False\rangle \in s \wedge \langle p, False\rangle \in s]$ (1)

$Happens(e, t) \leftrightarrow$ (2)

$[e = MoveP \wedge t = 1sec] \vee [e = MoveA \wedge t = 3sec] \vee [e = MoveB \wedge t = 4sec]$.

$Initiates(e, p, t) \leftrightarrow$ (3)
$[e = MoveP \wedge p = swing(PendulumP, F', \varphi, x_A, (10, 1sec, 0.3m)) \wedge t = 1sec]$
$\vee [e = MoveA \wedge p = engine(BallA, F, y_A, (0m, 0.4m/sec, 2sec)) \wedge t = 2sec]$
$\vee [e = MoveB \wedge p = engine(BallB, F, y_B, (4m, -0.3m/sec, 4sec)) \wedge t = 4sec]$.

By the rules (PS3) -- (PS6), a nearest event e_{AP} of the collision between the ball *A* and the pendulum *P* occurs at time 2.75*sec*, since the condition of the constraint in (PS5) -- (PS6) is satisfied. From (1), (2), (3) and (PS5) -- (PS6), it follows that in all models under circumscription we have

$(\exists s)[State(2.75sec, s) \wedge HoldsIn(engine(BallA, F, y_A, (0.3m, -0.4m/sec, 2.75sec)), s) \wedge HoldsIn(swing(PendulumP, F', \varphi, (10, 1sec, 0.3m)), s)]$.

It means that the pendulum *P* continues its motion in the same direction, and the ball *A* moves towards the opposite direction with its initial speed after the collision occurs after the collision. Furthermore,

$HoldsAt(moving(BallA, y_A, (0.3m, -0.4m/sec, 2.75sec)), t)$.
$HoldsAt(angle(PendulumP, \varphi, (10, 1sec, 0.3m)), t)$.

where $t \geq 2.75sec$.

3.3 Soundness and completeness

Let $D = (P, E)$ be consistent domain description for process semantics, where *P* is a set of initial processes and *E* is a set of events. We write $P = (p_1, p_2, ..., p_n)$ and $E = (e_1, e_2, ..., e_n)$.

Soundness Theorem Let *D* be a consistent domain description for process semantics and π denote the translation from the process semantics into the event calculus, for any process *P* if πD entails πP, then *D* entails *P*.

Completeness Theorem Let *D* be a consistent domain description for process semantics and π denote the translation from the process semantics into the event calculus, for any process *P* if *D* entails *P*, then πD entails πP.

4. LOGIC PROGRAMMING

We have implemented a logic programming system supporting the process semantics based on the event calculus in Prolog under the environment of Eclipse.

A logical program for the example of balls and pendulum is shown in Fig.2.

```
%primitive actions occurences %
happens (e(2), start_ball(a), 0.4, 0)
happens(e(4), start_ball(b), -0.3, 4)
happens(e(1), start_pendulum(p), 10, 1, 0.3)

%initial process specification %
initial_proc([ball(a), ball(b), pendulum(p), pos(a,0),
pos(b,4), v(a,0),
v(b,0), v(p,0), last_Tdc(0), t(a, 0), t(b, 0), t(p,0)]).

%specification of events and processes%
holdsAt(moving(ball(X), pos(X,Y), v(X,V), t(X,T), L),
TT):-
    holds(after(e(T), engine(ball(X), pos(X,Y), v(X,V),
t(X,T),L))),
    TT>T, T>2, L is Y + V*(TT-T).
holdsAt(angle(pendulum(p), pos(p, Y), var(p, V), t(p,
T), L), TT):-
   holds(after(e(T), swing(pendulum(p), pos(p, Y),
var(p, V), t(p, T), L))),
   TT>T, T> 1, L is –10* cos(2*3.14*(TT-T)).
%specification of state translation%
transition(Old, e(T), New) :- next_event(Old, e(T)),
next_proc(Old, e(T), New).
next_event(Old, e(Tdc)) :- proc_match([last_Tdc(T)],
          Old, _), trigger(Old, Tdc),Tdc>T.
trigger(_, T):- happens(e(T), _,_,_).
trigger(Old, Tdc):- proc_match([ball(X), pendulum(p),
                    v(X, VX), var(p, VY),
                    pos(X, X0), pos(p, Y0), t(X, TX),
                    t(p, TV)], Old,_),
                    Tdc is  TX+(Y0-X0)*VX.
```

Fig. 2. A Logical program for the example of pendulum and balls

5. SUMMARY

This paper presents an approach based on a high-level semantics of process to reason about continuous change. We believe that defining a well-formed semantics description is critical to solve the problem of reasoning in physical continuous system. With a case study we have shown how to integrate a high level semantics description with the

event calculus to reasoning about relative complex processes and continuous changes. We have proved the soundness and completeness of the event calculus with respect to the process and implemented the automated reasoning about processes in the logical programming framework.

The limitation of our approach based on the process semantics is not to consider events that occur simultaneously. Attempts to solve this problem have been made for reasoning about change in the discrete case [4, 14, 24, 17, 25]. In our approach, if some events occur at the same time but without mutual influences, this could be represented by our approach. But if two or more simultaneous events involve identical objects, then the overall results might not simply by the combination of the results of the involved events. This requires more sophisticated means to specify suitable state transitions. To extend our current approach to include simultaneous occurrences of events will be our future work.

References

[1]. Allen J. Toward a general theory of action and time. Artificial Intelligence, 1984, (23):123-154.

[2]. Kartha F, Lifschitz V. Soundness and completeness theorem for three formalization of actions. In Proc. International Joint Conference on Artificial Intelligence, France, 1993, pp 724-729.

[3]. Belleghem K, Denecker M, de Schreye D. Representing continuous change in the abductive event calculus. In Proc. International Conference on Logic Programming, 1995, pp 225-240.

[4]. Baral C, Gelfond M. Representing concurrent actions in extended logic programming. In Proc. International Joint Conference on Artificial Intelligence, France, 1993, pp 866-871.

[5]. Bibel W, Eder E. Methods and calculus for deduction. Handbook of Logic in Artificial Intelligence and Logic Programming, Clarendon Press, Oxford, 1993, (1): 68-184.

[6]. Grosskreutz, H., Lakermeyer,G.: ccGogol: A logical language dealing with continuous change. Logical Journal of IGPL 11 (2) (2003) 179—221

[7]. Gelfond M, Lifschitz V. Representing action and change by logic programs. Journal of Logic Programming, 1993, (17): 301-321.

[8]. Herrmann C, Thielscher M. Reasoning about continuous change. In Proc. of AAAI, Portland, U.S.A. 1996, pp639-644.

[9]. Kowalski R, Sadri F. Reconciling the situation calculus and event calculus. Journal of Logic Programming, 1997, (31): 39-58.

[10]. Kowalski R, Sergot M. A logic based calculus of events. New Generation of Computing, 1986, (4): 319-340.

[11]. Kowalski R. Database updates in the event calculus. Journal of Logic Programming, 1992, (12):121-146.

[12]. Li C. Reasoning about processes and continuous change. Shaker-Verlag, Aachen, 1999.

[13]. Lifschitz V. Circumscription. In The Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3: Non-monotonic Reasoning and Uncertain Reasoning (C.Hogger, D. Gabbay and J. Robinson eds.),1994, pp297-352.

[14]. Lin F, Shoham Y. Concurrent actions in the situation calculus. In Proc. AAAI, San Jose, U.S.A., 1992, pp590-595.

[15]. Miller R. A case study in reasoning about action and continuous change. In Proc. ECAI, Budapest, Hungary, 1996, pp624-628.

[16]. Miller R, Shanahan M. Reasoning about discontinuities in the event calculus. In Proc. 5th International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, Massachusetts, U.S.A., 1996, pp63-74

[17]. Reiter R. Natural actions, concurrency and continuous time in the situation calculus. In Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, Massachusetts, U.S.A., 1996, pp2-13.

[18]. Sadri F, Kowalski R. Variants of the event calculus. In Proc. International Conference on Logic Programming. MIT Press, 1995.

[19]. Sandewall E. The range of applicability and non-monotonic logics for the inertia problem. In Proc. International Joint Conference on Artificial Intelligence, France, 1993, pp738-743.

[20]. M. Shanahan. A circumscriptive calculus of events. Artificial Intelligence,1995, (77):249-284.

[21]. M. Shanahan. Representing continuous change in the event calculus. In Proc. ECAI 90, 1990, pp 98-113.

[22]. M. Shanahan. The event calculus explained. Lecture Notes in Artificial Intelligence 1600, Springer-Verlag, 1999, pp409-430.

[23]. Shoham Y, McDermott D. Problems in formal temporal reasoning. Artificial Intelligence 1988, (36):49-61

[24]. Thielscher M. Representing actions in equational logic programming. In Proc. International Joint Conference on Logic Programming, Italy, 1994, pp 207-224.

[25]. Thielscher M. The concurrent, continuous fluent calculus. Journal of Studia Logica, 2000, (67): pp 315—331