Automating proofs in event logic

Mark Bickford

Is the event logic formalism a good candidate for totally automated proofs?

Reasons to think so:

Essentially one type (events E) and two relations =, causal order Four standard skolem functions:

sender(e), last-change(x,e) -- backward in time

check-pre(a,e), receive-from(l,tg,e) -- forward in time

Using these, reasoning can be reduced to first-order logic??

Example: The "Once Lemma"

Realizable "event of kind locl(a) occurs exactly once at location i"

Realizer is

```
@i done:Bool initially = ff
@i precondition a(v:Unit) is done:Bool = ff
@i effect locl(a)(v:Unit) done := tt
@i only [locl(a)] affects done
```

C(e,e') == e causally before e'
loc(e) == location of e

axioms:

all a b c (C(a,b) -> (C(b,c) -> C(a,c))).

all a - C(a,a).

all a b (CE(a,b) <-> (C(a,b) | (a = b))).

all a b (CL(a,b) <-> (C(a,b) & (loc(a) = loc(b)))).

all a b (CEL(a,b) <-> (CE(a,b) & (loc(a) = loc(b)))).

all a b ((loc(a) = loc(b)) -> (C(a,b) | ((a = b) | C(b,a)))).

all a CEL(first(a),a).

@i precondition a(v:Unit) is done:Bool = ff

1(x) == !x
when1(e) == done when e
after1(e) == done after e
init1(i) == done initially @ i
lc1(e) == last-change(done,e)

all e ((loc(e) = i) -> ((kind(e) = locl(a)) -> - P1(when1(e)))).
all e ((loc(e) = i) ->
    (exists eprime (CEL(e,eprime) & ((kind(eprime) = locl(a))
      | - - P1(after1(eprime)))))).
- P1(init1(i)) -> (exists e (loc(e) = i)).

```
@i effect locl(a)(v:Unit) done := tt

all e ((loc(e) = i) -> ((kind(e) = locl(a)) -> P1(after1(e)))).

@i only [locl(a)] affects done

all e loc(e) = i ->  -(kind(e) = locl(a)) -> after1(e) =
    when1(e).

@i done:Bool initially = ff

-P1(init1(i)).
```

```
"axioms"
all e when1(first(e)) = init1(loc(e)).   "when-first-init"

"last-change"
all e ( all eprime ((CEL(eprime,e) -> (when1(eprime) =
    when1(e))) &
                    (CL(eprime,e) -> (after1(eprime) =
    when1(e)))))
| (CL(lc1(e),e) &
   -(after1(lc1(e)) = when1(lc1(e))) &
  (after1(lc1(e)) = when1(e)) &
  ( all eprime ((CL(lc1(e),eprime) -> (CL(eprime,e) ->
                 (when1(eprime) = when1(e)))) &
                (CEL(lc1(e),eprime) -> (CL(eprime,e) ->
                 (after1(eprime) = when1(e))))))
  ).
```

To show: event of kind locl(a) occurs exactly once at location i

"at most once"
all e eprime (loc(e)=i -> loc(eprime)=i ->
kind(e)=locl(a) -> kind(eprime)=locl(a) ->
e = eprime

"at least once"
exists e (loc(e)=i & kind(e)=locl(a))

problem in first-order logic with equality:

axioms & translations of constraints => goals

possible solvers: Jprover, otter, "guided" SAT, QBF?

Using otter for "at most once"

(loc(e) = i).

(loc(eprime) = i).

(kind(e) = locl(a)).

(kind(eprime) = locl(a)).

-(e = eprime).

C(eprime,e) | C(e,eprime).

From either, otter finds proof, but from the "or" it did not find the proof (Using it's "autonomous" mode, it stopped the search beacuse "sos empty")

Using otter for "at least once"

all e (loc(e)=i -> -(kind(e)=locl(a)).`

skolemize & instantiate "last-change":

```
loc(e0) = i.
CEL(e0,e).
(kind(e) = locl(a)) |  P1(after1(e)).
P1(after1(e)).
( all eprime ((CEL(eprime,e) -> (when1(eprime) = when1(e))) &
              (CL(eprime,e) -> (after1(eprime) = when1(e)))))
| (CL(lc1(e),e) &
  -(after1(lc1(e)) = when1(lc1(e))) &
  after1(lc1(e)) = when1(e) &
  all eprime ((CL(lc1(e),eprime) -> CL(eprime,e) ->
               when1(eprime) = when1(e)) &
               (CEL(lc1(e),eprime) -> CL(eprime,e) ->
                after1(eprime) = when1(e)))
  ).
```

Otter finds proof from either

all eprime ((CEL(eprime,e) -> (when1(eprime) = when1(e)))).

or its negation, but ??