CS 5860 Lecture Notes: A Theory of Events in First Order Logic (November 15, 2011)

1 Introduction

We have seen how to express a few protocols in the setting of computation by *asynchronous message passing*, such as mutual exclusion, message acknowledgement, and collecting responses to liveness "pings". We also presented an intuitive "theory of events" at a finite number of locations $\texttt{loc}_1, \ldots, \texttt{loc}_n$ which send and receive messages. We stressed the idea that there is no global notion of "time" (no global clock) and that we reason about time in terms of Lamport's notion of *causal order* among events.

Now we will see how to express these concepts about events in first-order logic. Unlike the case for first-order number theory where the domain of discourse D is the type of natural numbers, for a theory of events, we need to subdivide the domain into several *sorts*. We start with the sort of *events* and *locations* defined by decidable predicates $\texttt{E}(x)$, $\texttt{Loc}(x)$ on D. For convenience, we also use $\texttt{Bool}(x)$ for Booleans and $\texttt{Unit}(x)$ for a sort with one object written as $\bullet$; we also have the natural numbers $\texttt{N}(x)$ when we want them. In the future we will need $\texttt{ID}(x)$ for identifiers, $\texttt{Link}(x)$ for communication links, $\texttt{Value}(x)$ for message values and so forth. All of these predicates simply divide D into separate *sorts*, all disjoint and *decidable*, i.e. $\forall x.(\texttt{E}(x) \vee \sim \texttt{E}(x))$, $\forall x.(\texttt{Loc}(x) \vee \sim \texttt{Loc}(x)), \forall x.(\texttt{N}(x) \vee \sim \texttt{N}(x))$, etc. Later we will see that *type theory* offers a richer and more flexible way to handle sorts and logic in a uniform way.

Recall the picture of our model of computation. This picture is sometimes called a *message sequence diagram*.

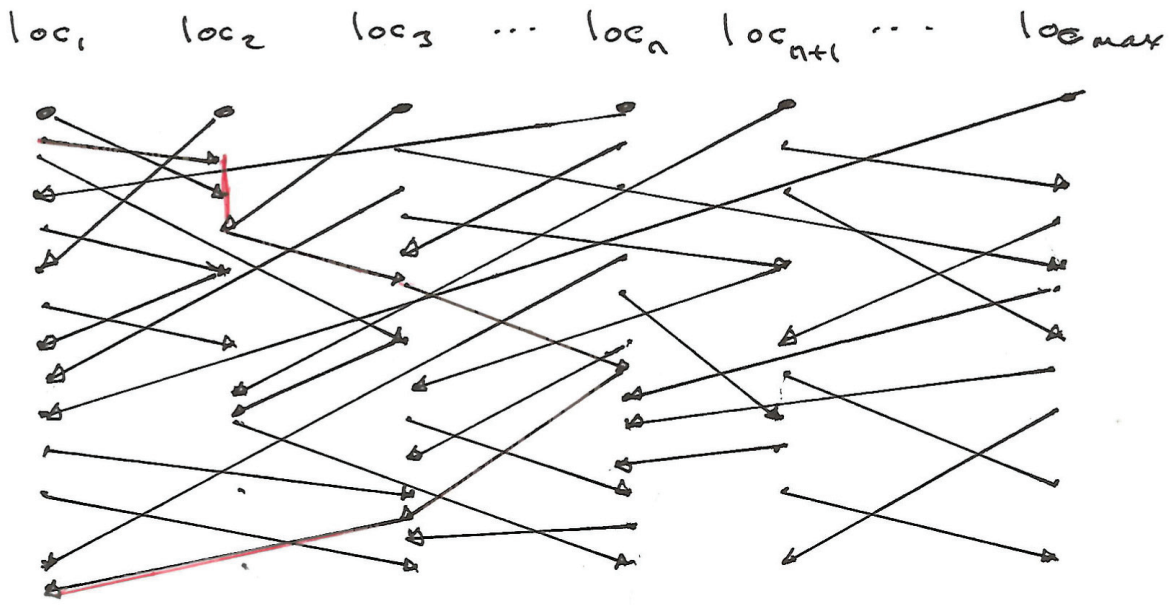

Fig. 1. message sequence diagram

At each location $\texttt{loc}_i$ events are *linearly ordered*, creating a *sequential notion of time* in which events are totally ordered. It does not make sense to draw an arrow going back in time at a location, and causal order proceeds downward as illustrated by the red links from $\texttt{loc}_1$ to $\texttt{loc}_2$ to $\texttt{loc}_3$ to $\texttt{loc}_n$ back to $\texttt{loc}_3$ back to $\texttt{loc}_1$. We show slow processes at a location by the fact that events are widely spaced, not by directing arrows backwards. Using backwards pointing arrows can lead to inadvertent temporal paradox.

We will capture properties of events caused by *computational processing* executing at each location, so we also think of the locations as *processes*. These processes could be executing many *threads* of computation distinguished by the *kind* of events.

Equality of events. We assume that equality on Locations and Events is decidable, thus
$\forall x, y.(E(x)\ \&\ E(y) \Rightarrow (x = y) \vee \sim (x = y))\quad \forall x, y.(\texttt{Loc}(x)\ \&\ \texttt{Loc}(y) \Rightarrow (x = y) \vee \sim (x = y))$

Notation. It is convenient to write *typed quantifiers* to express the above concepts as well as many others, e.g. $\forall x, y : E.(x = y \vee \sim (x = y))$ and $\forall x : E.\forall i : \texttt{Loc}.P(x, i)$ means $\forall x, i.(E(x)\ \&\ \texttt{Loc}(i) \Rightarrow P(x, i))$.

2 Event orderings

Axiom

$\forall x : E.\exists y.(E(y)\ \&\ \texttt{Predecessor}(x, y)) \vee (\texttt{Loc}(y)\ \&\ \texttt{occurs\_at}(x, y))$

The *realizer* for this axiom is the term $\texttt{pred?}(x)$, a computable term that defines a function on events whose value is a pair $< y, p >$ where y is in the domain D and p is either $\texttt{inl(*)}$ or $\texttt{inr(*)}$ [1]

Given a particular event e at a location $\texttt{loc}_i$, the term $\texttt{pred?}(x)$ will decide whether e is the initial event at $\texttt{loc}_i$ and if so, it will return $< \texttt{loc}_i, \texttt{inr}(*) >$. Otherwise, $\texttt{pred?}(e)$ will compute to the previous event at the location

By examining the second component of $< y, p >$ we can tell whether the result is an event, e.g. p is $\texttt{inl}(x)$, or a location.

If e is not the initial event, then we can examine the sequence of events at the location of e and find its predecessor. To do this, we need to be able to compute the *location* of the event.

Axiom

$\forall x : E.\exists y : \texttt{Loc}.\ \texttt{Occurs\_at}(x, y)$

The realizer is a term $\texttt{loc}(x)$ which for any event x computes the unique location of the event. For simplicity we let $\texttt{loc}(x)$ have value i rather than $< i, * >$. The other option would be to have $\texttt{locof}(x) =< i, * >$ and $\texttt{loc(x)} = \texttt{spread}(\texttt{locof}(x); i, a.i)$. All events happen at a process location, and we postulate some unspecified mechanism to find the location. In the formal mathematical model we simply define an event to include its location, e.g. in one formal model from 2003 an event is a pair $< i, t >$ where i is the location and t is a discrete time step.

[1] Recall that evidence for atomic predicates such as $\texttt{E}(y)$ and $\texttt{Loc}(y)$ is often just a token, *.

Next we need an axiom for finding the sender of an event. If the event is not a receive, then we associate the unit value rather than the sender.

Axiom

$\forall x : E.\exists y.(E(y)$ & $\texttt{Sender}(x, y)) \vee (\texttt{Unit}(x)$ & $\texttt{NotReceive}(x))$

The realizer is the term $\texttt{sender?}(x)$. The term computes by finding the canonical form of the event. If it has the form $\texttt{rcv}(v)$ then we find the sender from the header or the channel (as with $\texttt{pred?}$). If x is not a receive, then $\texttt{sender?}(x)$ computes to the unit value $\bullet$.

Now we will define $x \lhd y$, also written as the binary relation $\texttt{Pred}(x, y)$. First we define these functions and predicates.

```
first?:E → Bool
first?(x)  =  spread(pred?(x);y,p. decide(p;l. false; r. true))
           =  let pred?(x) = (y,p) in if isl(p) then false
                                                else true
sender?: E → E + Unit
  rcv?(x)  =  decide(sender?(x); l. true; r. false)
           =  if isl(sender?(x)) then true else false.
 First(x) iff first?(x) = true
   Rcv(x) iff rcv?(x) = true
Pred(x,y) iff (¬ First(y) & x = pred(y))
              ∨ (Rcv(y) & x = sender(y))
thus on y such that ¬ First(y),
pred(y) = spread(pred?(y);x,p.x)
and on y such that rcv(y),
sender(y) = if isl(sender?(y)) then outl(y)                    .
```

We will now form the *transitive closure* of $\texttt{Pred}(x, y)$, this will be Lamport's *causal order* relation, $x < y$. We will be able to prove $\forall x, y : E.((x < y) \vee \sim (x < y))$, but we need more axioms.

Given a relation $\texttt{R}(x, y)$ we define its *transitive closure* as follows. Define $\texttt{R}^{(0)}(x, y)$ $\texttt{iff}$ $\texttt{R}(x, y)$ and $\texttt{R}^{(n+1)}(x, y)$ $\texttt{iff}$ $\texttt{R}(x, z)$ & $\texttt{R}^{(n)}(z, y)$ for some z.

$$R^*(x, y) \texttt{ iff } \exists n : \mathbb{N}.R^{(n)}(x, y).$$

Using the notation $x \lhd y$ for $\texttt{Pred}(x, y)$, here is the same definition. Define $x \lhd^{(0)} y$ $\texttt{iff}$ $x \lhd y$ and $x \lhd^{(n+1)} y$ $\texttt{iff}$ $\exists z.(x \lhd z$ & $z \lhd^{(n)} y.)$.

Say $x \lhd^* y$ $\texttt{iff}$ $\exists n : \mathbb{N}.x \lhd^{(n)} y$.

Definition: Lamport's *causal order on events* is $\texttt{Pred}^*$ (same as $\lhd^*$).

We will show that we can reason by induction on $\texttt{Pred}^*$. An elegant way to do this is by postulating that $\texttt{Pred}(x, y)$ is strongly well founded.

Axiom

$\texttt{Pred}(x, y)$ is *strongly well founded*, i.e. there is a "choice sequence f" from $\texttt{E}$ to $\mathbb{N}$ such that $\forall e, e' : E.\ \texttt{Pred}(e, e') \Rightarrow f(e) < f(e')$.

We also need an axiom about $\texttt{pred}(x)$.

Axiom

The predecessor function, $\texttt{pred}$, is injective (i.e. one-to-one).
$\forall e, e\prime : E.(\texttt{loc}(e) = \texttt{loc}(e') \ \& \ \neg\texttt{First}(e) \ \& \ \neg\texttt{First}(e')) \Rightarrow$
$(\texttt{pred}(e) = \texttt{pred}(e')) \Rightarrow e = e'$.

Theorem $\texttt{Pred}^*(x, y)$, causal order, is strongly well founded.