Concurrent Algorithms in Linear Logic Programming

Robert J. Simmons (Work with Frank Pfenning)

February 14, 2008

Concurrent Algorithms in Linear Logic Programming

Robert J. Simmons (Work with Frank Pfenning)
February 14, 2008

Linear Logical Algorithms. Robert J. Simmons and Frank Pfenning. Submitted, February 2008.
(available from http://www.cs.cmu.edu/~fp)

Inference Rules

$$\frac{a \quad b}{c}$$

In Meld, Datalog, XSB:

“c :- a,b.”

Inference Rules

$$\frac{a \ \textit{true} \qquad b \ \textit{true}}{c \ \textit{true}}$$

Inference Rules

$$\frac{\text{path}(X,Y)\ \textit{true} \qquad \text{path}(Y,Z)\ \textit{true}}{\text{path}(X,Z)\ \textit{true}}$$

Inference Rules

$$\frac{\begin{array}{l}\text{path}(X,Y)\\ \text{car\_is\_at}(C,X)\end{array}}{\text{car\_is\_at}(C,Y)}$$

Inference Rules

$$\frac{\text{path}(X,Y)\ \textit{true} \qquad \text{car\_is\_at}(C,X)}{\text{car\_is\_at}(C,Y)}$$

Inference Rules

$$\frac{\text{path}(X,Y)\ \textit{true} \qquad \text{car\_is\_at}(C,X)\ ?}{\text{car\_is\_at}(C,Y)}$$

Inference Rules

$$\frac{\text{path}(X,Y)\ \textit{true} \qquad \text{car\_is\_at}(C,X)\ \textit{?}}{\text{car\_is\_at}(C,Y)\ \textit{?}}$$

Inference Rules

$$\frac{\text{path}(X,Y)\ \textit{persistent} \qquad \text{car\_is\_at}(C,X)\ \textit{?}}{\text{car\_is\_at}(C,Y)\ \textit{?}}$$

Inference Rules

$$\frac{\text{path}(X,Y)\ \textit{persistent} \qquad \text{car\_is\_at}(C,X)\ \textit{ephemeral}}{\text{car\_is\_at}(C,Y)\ \textit{?}}$$

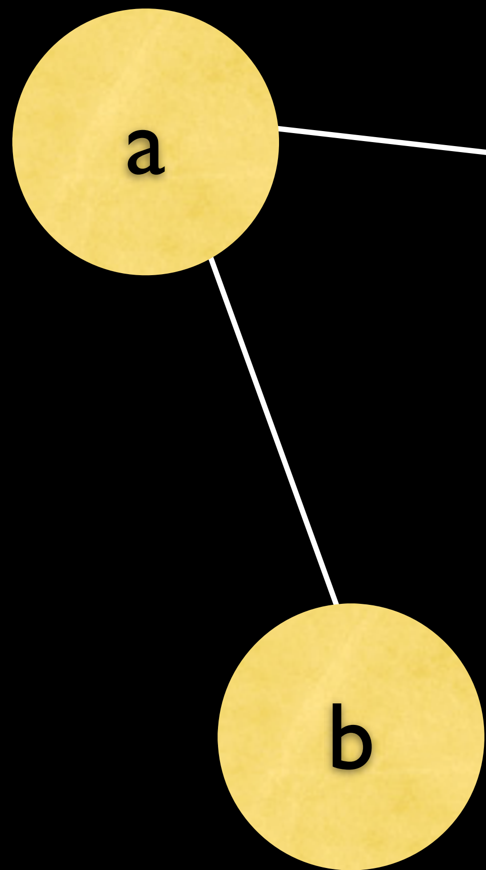
Inference Rules

$$\frac{\mathrm{path}(X,Y)\ \textit{persistent} \qquad \mathrm{car\_is\_at}(C,X)\ \textit{ephemeral}}{\mathrm{car\_is\_at}(C,Y)\ \textit{ephemeral}}$$

Inference Rules

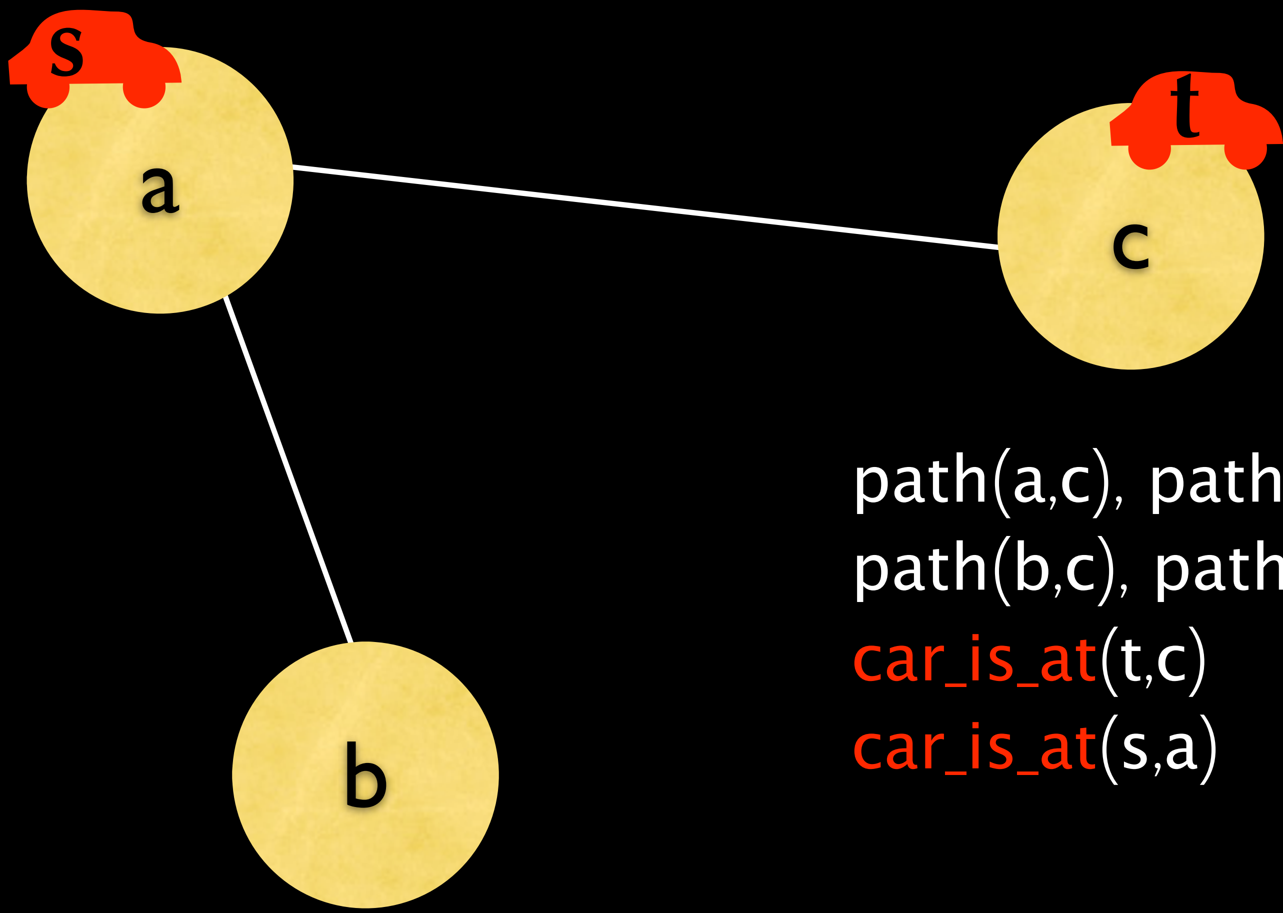
$$\frac{\begin{array}{l}\text{path}(X,Y)\ \textit{persistent} \\ \text{car\_is\_at}(C,X)\ \textit{ephemeral}\end{array}}{\text{car\_is\_at}(C,Y)\ \textit{ephemeral}}$$

(Linear Logic!)



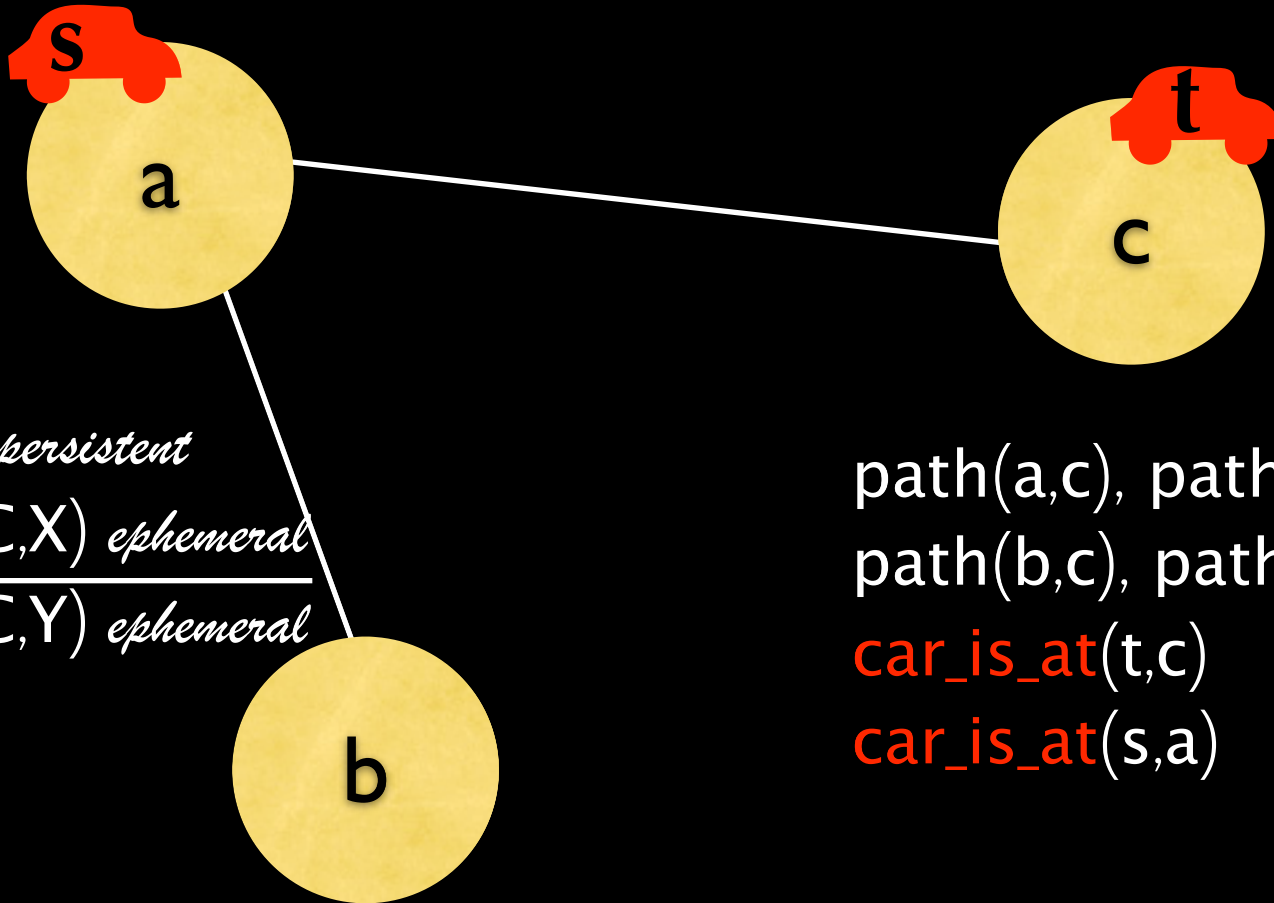
path(a,c), path(c,a), path(a,b),
path(b,c), path(c,b), path(b,a)

Persistent propositions
describe permanent truth the system



path(a,c), path(c,a), path(a,b),
path(b,c), path(c,b), path(b,a)
car_is_at(t,c)
car_is_at(s,a)

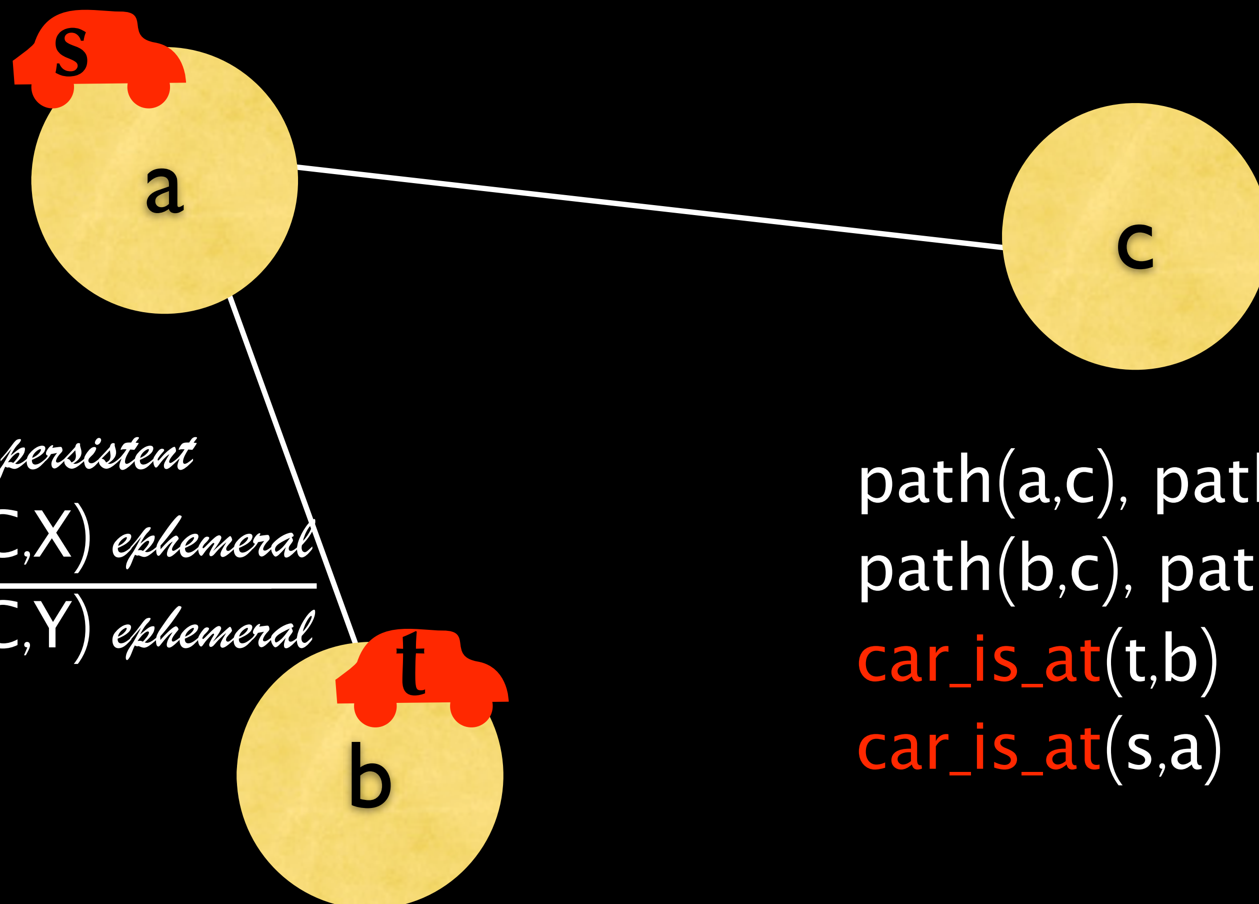
Ephemeral propositions
describe current state of system



$$\frac{\text{path}(X,Y)\ \textit{persistent} \qquad \text{car\_is\_at}(C,X)\ \textit{ephemeral}}{\text{car\_is\_at}(C,Y)\ \textit{ephemeral}}$$

path(a,c), path(c,a), path(a,b),
path(b,c), path(c,b), path(b,a)
car_is_at(t,c)
car_is_at(s,a)

Ephemeral rules describe potential changes to system



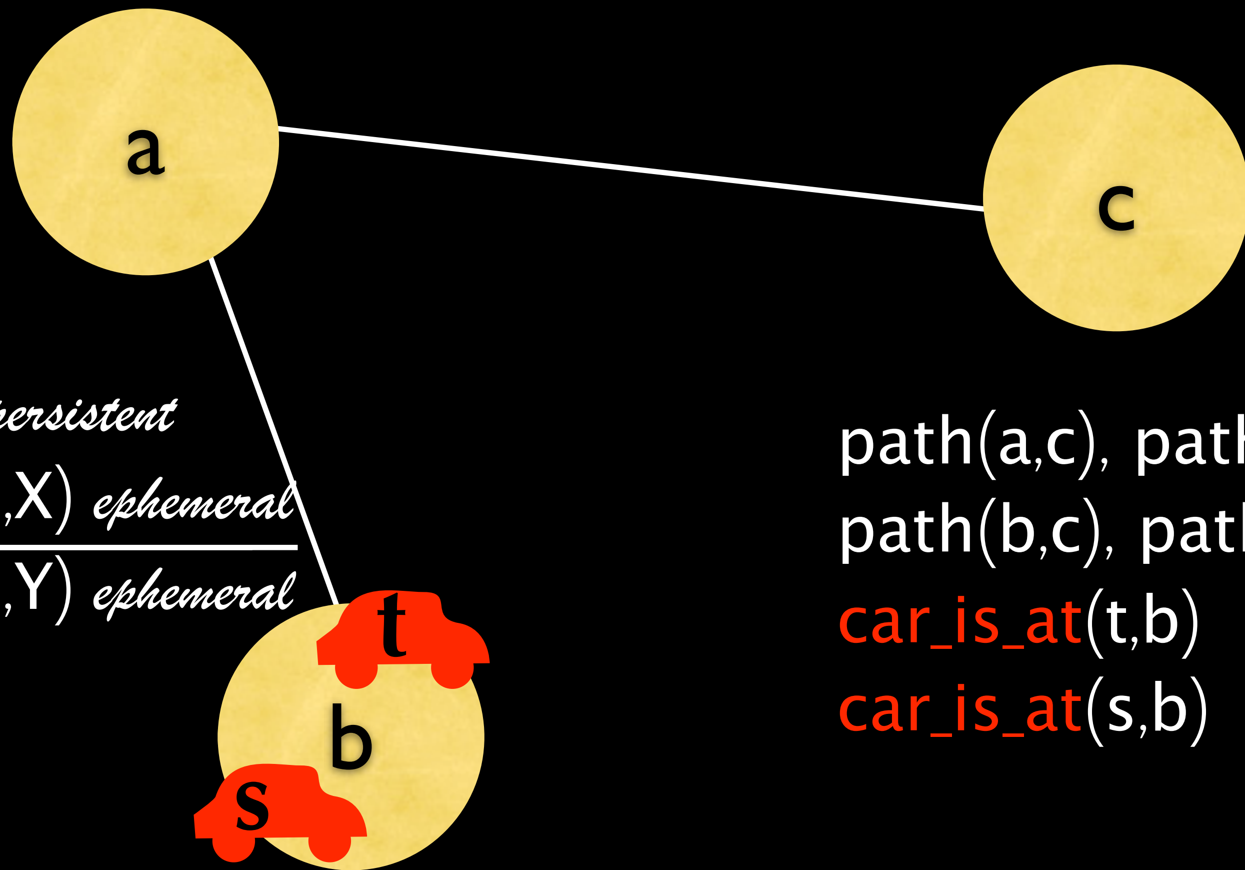
path(X,Y) *persistent*
car_is_at(C,X) *ephemeral*

---

car_is_at(C,Y) *ephemeral*

path(a,c), path(c,a), path(a,b),
path(b,c), path(c,b), path(b,a)
car_is_at(t,b)
car_is_at(s,a)

Ephemeral rules describe potential changes to system



$$\frac{\begin{array}{l}\text{path}(X,Y)\ \textit{persistent}\\ \text{car\_is\_at}(C,X)\ \textit{ephemeral}\end{array}}{\text{car\_is\_at}(C,Y)\ \textit{ephemeral}}$$

path(a,c), path(c,a), path(a,b),
path(b,c), path(c,b), path(b,a)
car_is_at(t,b)
car_is_at(s,b)

Ephemeral rules
describe potential changes to system

$$\frac{\text{unvisited}(\text{root})}{\text{visited}(\text{root})}$$

$$\frac{\begin{array}{c}\text{edge}(X,Y)\\ \text{visited}(X)\\ \text{unvisited}(Y)\end{array}}{\begin{array}{c}\text{visited}(Y)\\ \text{tree}(X,Y)\end{array}}$$

Ephemeral propositions are "mutual exclusion locks"

$$\frac{\text{unvisited(root)}}{\text{visited(root)}}$$

$$\frac{\text{edge(X,Y)} \quad \text{visited(X)} \quad \text{unvisited(Y)}}{\text{visited(Y)} \quad \text{tree(X,Y)}}$$

$$\frac{\mathrm{item}(X) \qquad \mathrm{list}(L)}{\mathrm{list}(\mathrm{cons}(X,L))}$$

Ephemeral propositions are "state objects"

$$\frac{\text{item}(X) \quad \text{list}(L)}{\text{list}(\text{cons}(X,L))}$$

Logical account of concurrency

Ephemeral propositions are
“mutual exclusion locks”
“state objects”
...what else?

Represent locality? (@modal logic)
Global synchronization? (temporal logic)

Declarative model of programming can allow more optimizations

Executable specifications!

Huge success with purely-persistent Datalog programming (Whaley et. al)

Spiral too, it seems!

Both are domain-specific (whether de facto or de jure)