

Using Time in Loom

Thomas A. Russ

USC

Information Sciences Institute

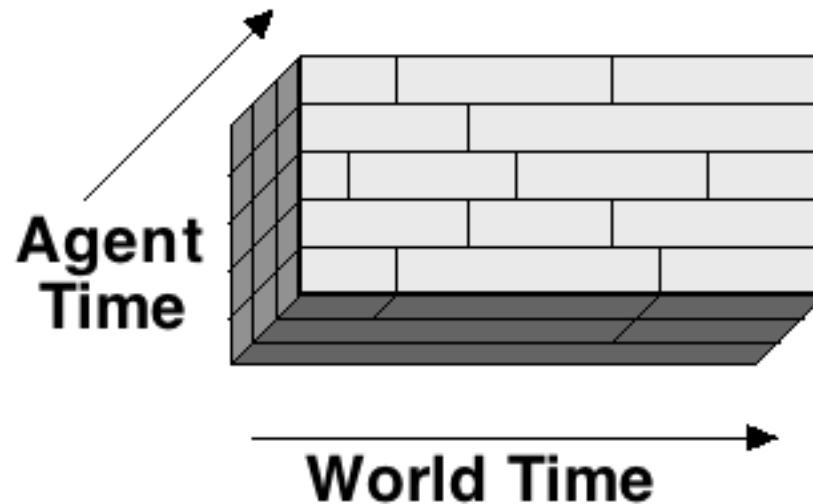


Outline

- *Time Representation*
- *Basic Assertions*
- *Basic Queries*
- *Persistence*
- *Time and the Classifier*
- *Advanced Examples*



Agent and World Time



- *World Time Records Domain Facts*
- *Agent Time Records Knowledge Base Changes*

Time Representation

■ *Definite Times*

- *Integers*
- *Time Strings “10/28/94 11:33”*

■ *Anchored to Calendar*

- *Common Lisp universal time*

■ *Points Are Basic Units*

■ *Intervals Are Derived*

■ *“Property” Interpretation of Intervals*



Properties and Events

■ *Properties*

- *True over all subintervals*
- *“The house is red”*

■ *Events*

- *True only over the entire interval*
- *“John ran completely around the track.”*



Basic Assertions

■ *Transitions Only*

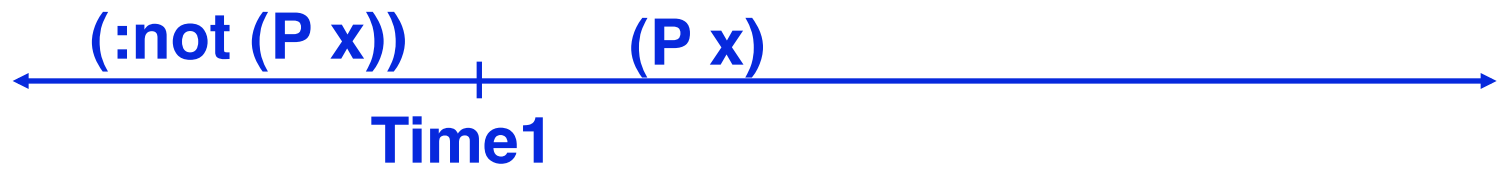
- *(:begins-at time-point assertion)*
- *(:ends-at time-point assertion)*

■ *Strong Temporal Assertion*

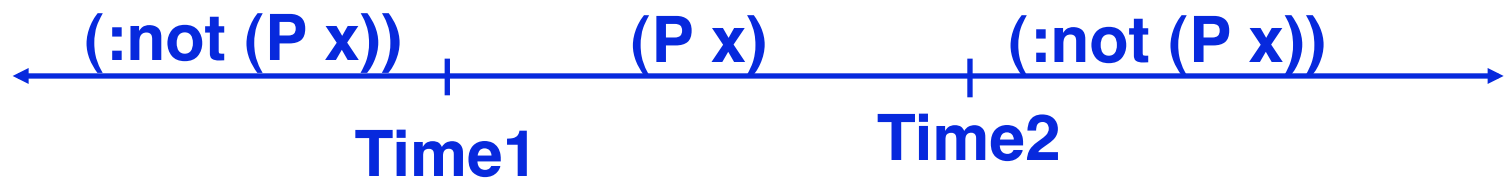
- *Before :begins-at, assertion is false.*
- *After :begins-at, assertion is true.*



Basic Assertions



`(tell (:begins-at Time1 (P x)))`



`(tell (:ends-at Time2 (P x)))`

Basic Queries—Transitions

■ *Transitions:*

- *(ask (:ends-at t1 (P x)))*



USC
ISI

■ *Transitions:*

- ***(ask (:ends-at t1 (P x)))***

■ **States:**

- *(ask (:holds-at t1 (P x)))*

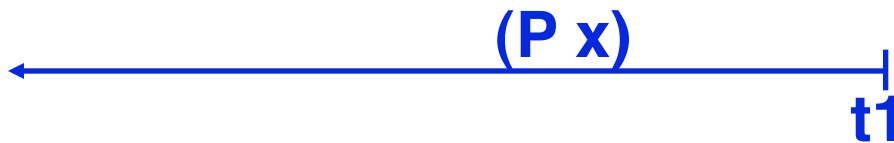
Basic Queries—States Problem

■ *Transitions:*

- *(ask (:ends-at t1 (P x)))*

■ *States:*

- *(ask (:holds-at t1 (P x)))*
- *But this can be ill-defined*



Basic Queries—States Solution

■ Introduce Directional Operators

- `(ask (:holds-before t1 (P x)))`
- `(ask (:holds-after t1 (P x)))`

■ Yields well-defined results:

$(P\ x)$

←—————|
t1

<code>:holds-before</code>	<code>==></code>	<code>t</code>
<code>:holds-after</code>	<code>==></code>	<code>nil</code>



Non-Transitional Assertions

■ *Persistence Only*

- *(:holds-after time-point assertion)*
- *(:holds-before time-point assertion)*

■ *Weak Temporal Assertion*

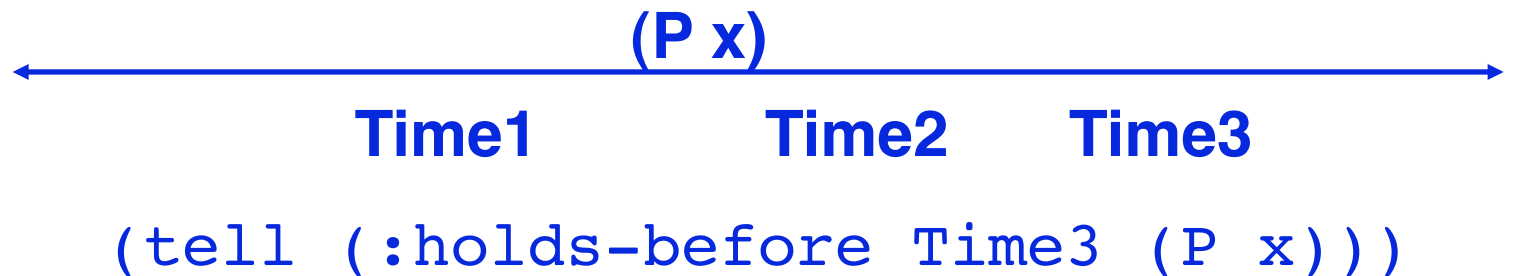
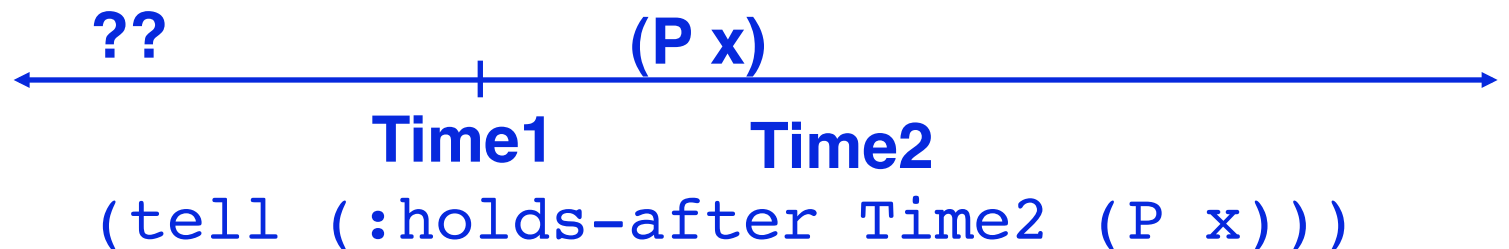
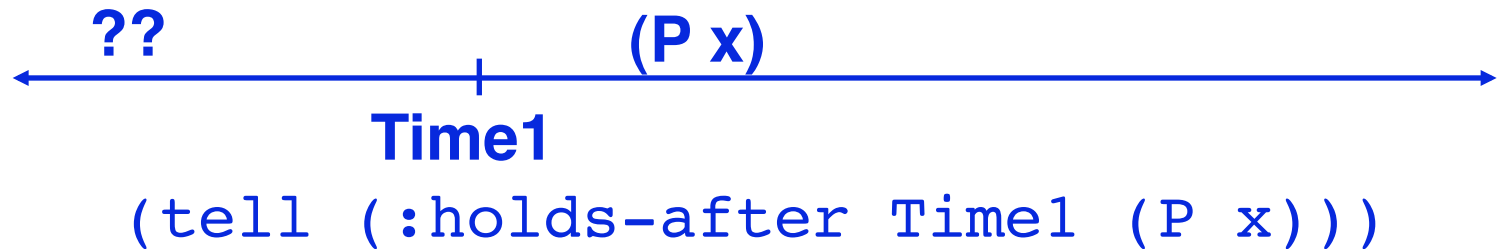
- *Before :holds-after, assertion can be true or false.*
- *After :holds-before, assertion can be true or false.*

■ *:holds-at is the combination of :holds-before and :holds-after*

- *The assertion is true both before and after a :holds-at*



Persistence Assertions



Temporal Operator Truth Table



	$(P\ x)$		
	t1	t2	t3
■ <i>:begins-at</i>	t	nil	nil
■ <i>:holds-after</i>	t	t	nil
■ <i>:holds-at</i>	nil	t	nil
■ <i>:holds-before</i>	nil	t	t
■ <i>:ends-at</i>	nil	nil	t

Changes to Classifier

■ *Classifier Is Time Sensitive*

- *Temporal information in the ABox affects classification*



■ *Definitions Are Time Invariant*

- *TBox definitions hold over the entire time line*

Bachelor Example

```
(defconcept Married  
  :characteristics :temporal)
```

```
(defconcept Bachelor :is  
  (:and Male (:not Married)))
```

```
(tell (Male p1)  
  (:begins-at t1(Married p1)))
```

(Male p1)

(:not (Married p1)) | (Married p1)

t1

(Bachelor p1)

t1

Widow Example

```
(defconcept Dead
  :characteristics :temporal)
```

```
(defrelation husband
  :is (:and spouse (:range Male))
  :characteristics :temporal)
```

```
(defconcept widow
  :is (:and Female
        (:some husband Dead)))
```



Widow Assertions

```
(tellm (Female Mary) (Male John))
```

```
(tellm (:begins-at "1/1/90"  
                  (spouse Mary John))  
       (:begins-at "1/1/94"  
          (Dead John)))
```

(Male John)

(Female Mary)

(spouse Mary John)

(Dead John)

1/1/90

1/1/94



Widow Derivation

```
(tellm (Female Mary) (Male John))
```

```
(tellm (:begins-at "1/1/90"  
                  (spouse Mary John))  
      (:begins-at "1/1/94"  
        (Dead John)))
```

(Male John)

(Female Mary)

(spouse Mary John)

(Dead John)

(Widow Mary)

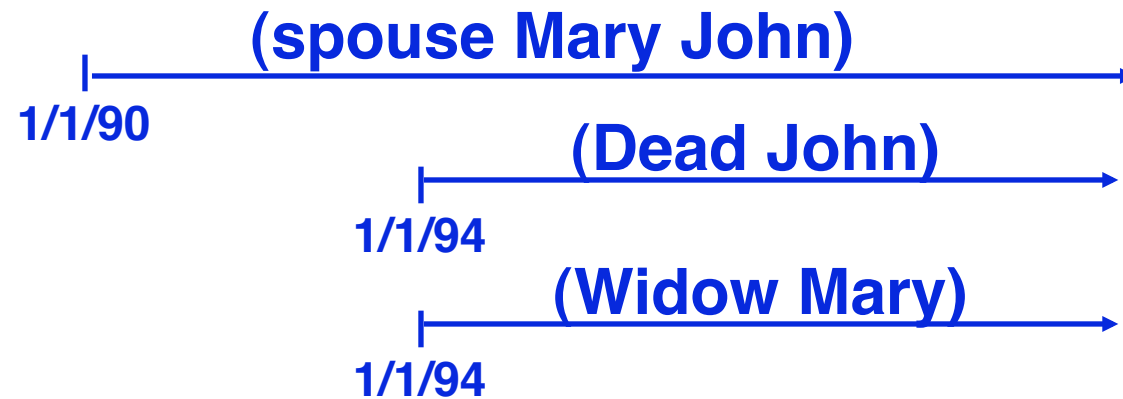
1/1/90

1/1/94

1/1/94



Widow Queries



```
(retrieve ?x (:holds-at "10/28/94"  
                        (widow ?x)))
```

```
=> (|i|Mary)
```

```
(retrieve ?x (:begins-at ?x  
                        (Widow Mary)))
```

```
=> (2966400000)  
; = "1/1/94 00:00:00"
```


Former Hockey Player

```
(defconcept former-hockey-player :is
  (:and person
    (:satisfies (?p)
      (:for-some (?t)
        (:and (past ?t)
          (:ends-at ?t
            (hockey-player ?p
              ))))))))
```



Former Hockey Player

```
(defconcept former-hockey-player :is
  (:and person
    (:satisfies (?p)
      (:for-some (?t)
        (:and (past ?t)
          (:ends-at ?t
            (hockey-player ?p
              ))))))))
```



- Temporal concept “**past**” constrains matches for ?t to occur before the time this definition is satisfied.
- A former hockey player is “someone who ceased to be a hockey player sometime in the past.”



Former Hockey Player

Temporal Clause

```
(defconcept former-hockey-player :is
  (:and person
    (:satisfies (?p)
      (:for-some (?t)
        (:and (past ?t)
          (:ends-at ?t
            (hockey-player ?p
              ))))))))
```

- *Temporal relation to the concept “hockey-player” established.*



Former Hockey Player Assertion and Queries

```
(tellm (Person Fred))
```

```
(tellm (:ends-at "1/1/90"  
                (hockey-player Fred)))
```



```
(ask (:holds-at "1/1/88"  
               (hockey-player Fred))) => T
```

```
(ask (:holds-at "1/1/88"  
       (former-hockey-player Fred))) => NIL
```

```
(ask (:holds-at "1/1/94"  
       (hockey-player Fred))) => NIL
```

```
(ask (:holds-at "1/1/94"  
       (former-hockey-player Fred))) => T
```


Summary

- *World and Agent Time Supported*
- *Definite, Calendar-Anchored Time*
- *ABox Supports Temporal Assertions*
- *Inference Is Time Sensitive*

