

## Asynchronous Circuit Design

Chris J. Myers

Lecture 7: Timed Circuits  
Chapter 7

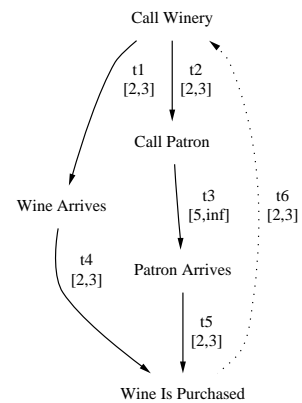
## Timed Circuits

- Previous methods only use limited knowledge of delays.
- Very robust systems, but extremely conservative.
- Large functional units do not have zero delay.
- Gates and wires do not have an infinite delay.
- Timing analysis can identify additional unreachable states.
- These unreachable states are additional don't cares.
- *Timed circuits* use this information to optimize the design.

## A Simple Example

- Shopkeeper actively calls winery and patron.
- Calls the patron immediately after calling the winery without waiting for the wine to arrive.
- The shopkeeper does the following:
  - Calls the winery,
  - Calls the patron,
  - Peers out the window until he sees both the wine delivery boy and the patron,
  - Lets them in, and
  - Completes the sale.

## Timing Relationships



## Timed States

- There is a timer  $t_i$  associated with each arc in the graph.
  - A *timed state* is an *untimed state* and value of all active *timers*.
- $$(\{r_6\}, t_6 = 0)$$
- A timer is allowed to advance by any amount less than its upper bound resulting in a new timed state.

$$(\{r_6\}, t_6 = 1.1)$$

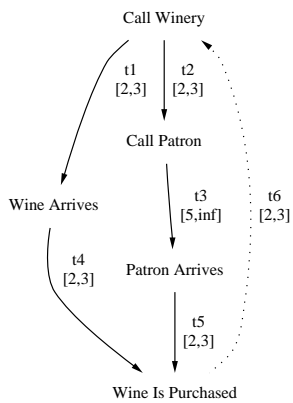
$$(\{r_6\}, t_6 = 2.22)$$

$$(\{r_6\}, t_6 = 2.71828182846)$$

## Timing Sequences

- When a timer reaches its lower bound, it becomes *satisfied*.
- When a timer reaches its upper bound, it becomes *expired*.
- An event enabled by a single rule must happen sometime after its timer becomes satisfied and before it becomes expired.
- When an event is enabled by multiple rules, it must happen after all of its rules are satisfied, but before all of its rules are expired.
- Extend the notion of allowed sequences to timed states paired with the time of the state transition.
- State transition can be either time advancement or a change in the untimed state.

## Example: Timing Sequence



- ( $\{r_6\}, t_6 = 0, 0$ ),
- ( $\{r_6\}, t_6 = 2.22, 2.22$ ),
- ( $\{r_1, r_2\}, t_1 = t_2 = 0, 2.22$ ),
- ( $\{r_1, r_2\}, t_1 = t_2 = 2.1, 4.32$ ),
- ( $\{r_1, r_3\}, t_1 = 2.1, t_3 = 0, 4.32$ ),
- ...

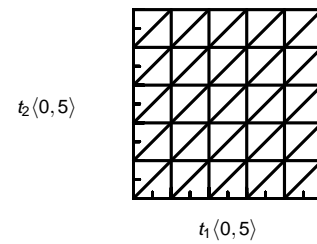
## Timed State Space Exploration

- Since time can take on any real value, there is an uncountably infinite number of timed states and timed allowed sequences.
- Must either group timed states into finite number of equivalence classes or restrict the values of the timers.
- Several possible methods for *timed state space exploration*:
  - Region method
  - Discrete-time method
  - Zone method
  - POSET method

## Regions

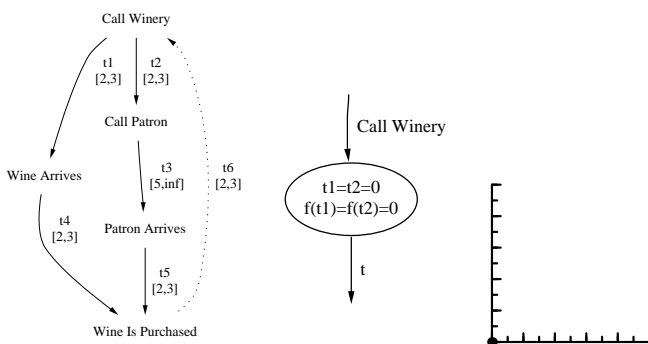
- A *region* is described by the integer component of each timer and the relationship between the fractional components.
- $f(t_1) = f(t_2) = 0$ : region is a point.
- $f(t_1) = 0$  and  $f(t_2) > 0$ : region is a vertical line segment.
- $f(t_1) > 0$  and  $f(t_2) = 0$ : region is a horizontal line segment.
- $f(t_1) = f(t_2) > 0$ : region is a diagonal line segment.
- $f(t_1) > f(t_2) > 0$ : region is an lower triangle.
- $f(t_2) > f(t_1) > 0$ : region is an upper triangle.

## Possible Timed States Using Regions

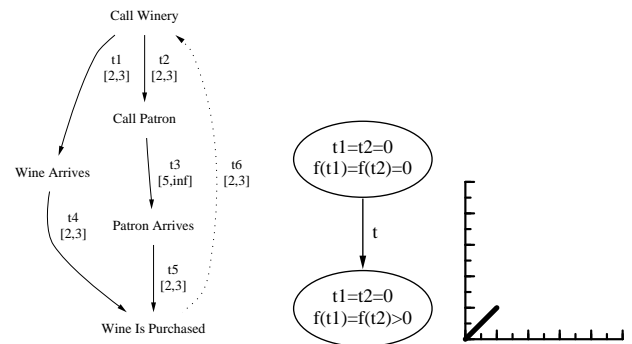


171 distinct timed states

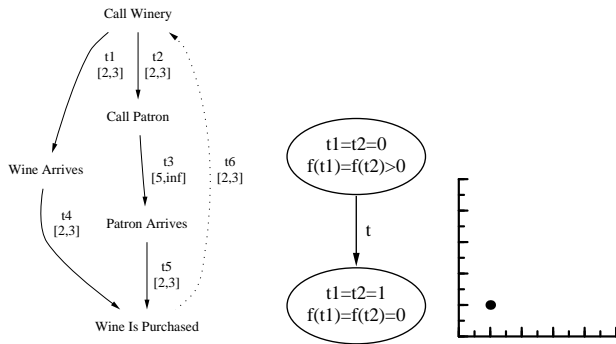
## Timed Sequence Using Regions



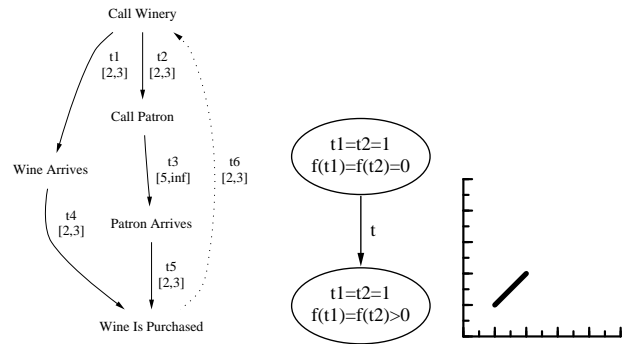
## Timed Sequence Using Regions



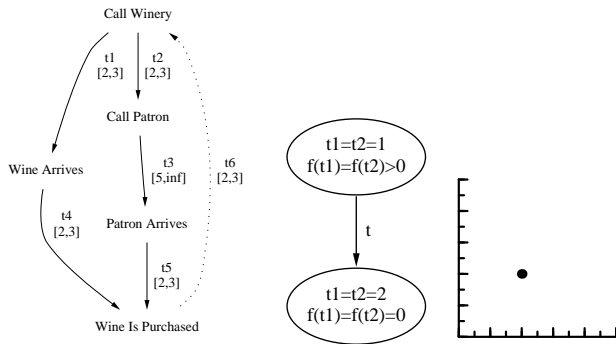
### Timed Sequence Using Regions



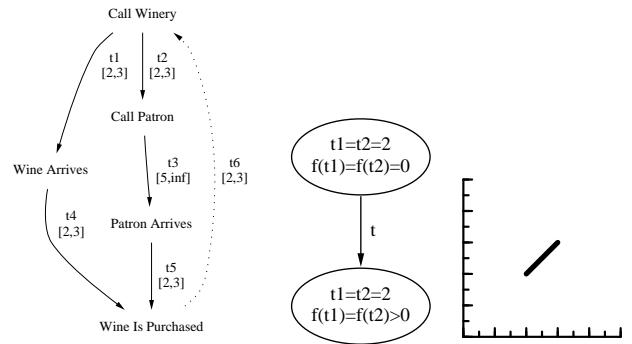
### Timed Sequence Using Regions



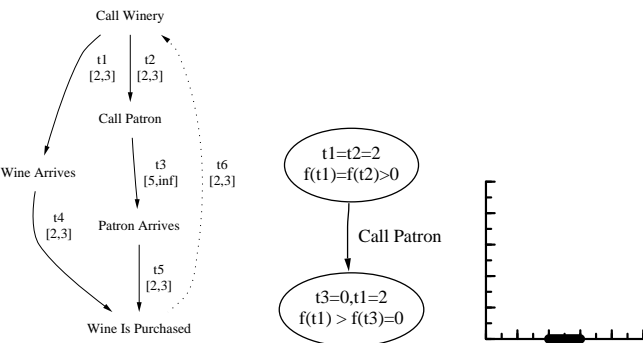
### Timed Sequence Using Regions



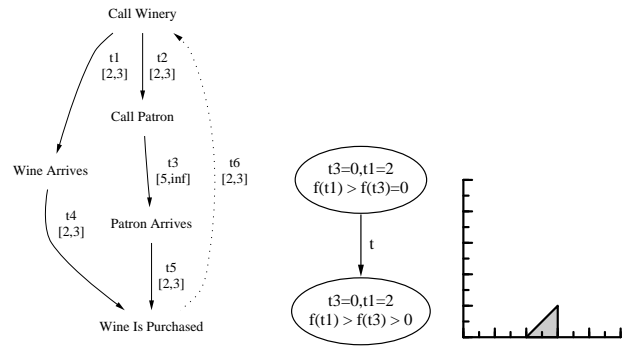
### Timed Sequence Using Regions



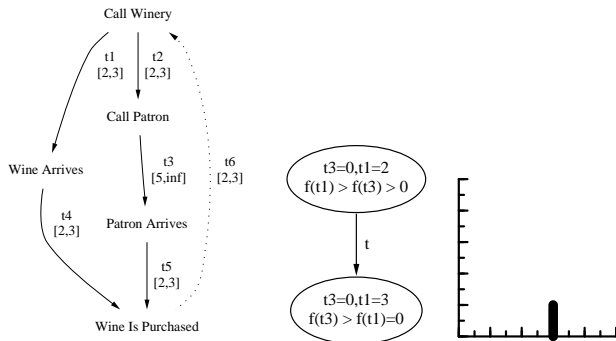
### Timed Sequence Using Regions



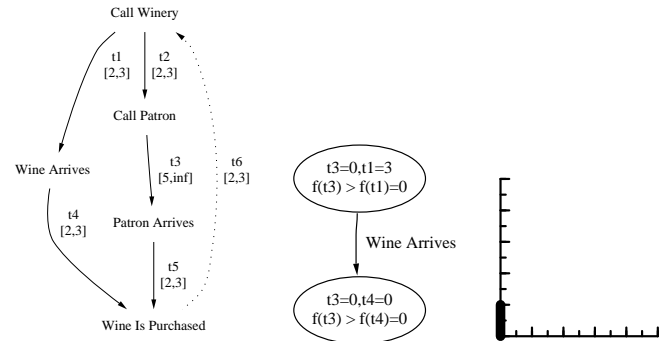
### Timed Sequence Using Regions



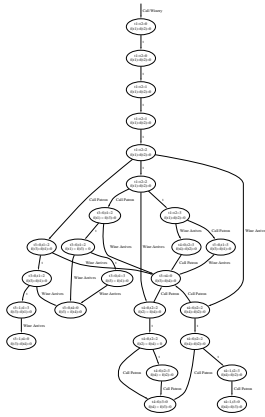
## Timed Sequence Using Regions



## Timed Sequence Using Regions



## Timed State Space Using Regions



## Timed State Space Explosion

- Requires 26 timed states to represent all the timing relationships for only 4 untimed states.
- Worst-case complexity is:

$$|S| \frac{n!}{ln2} \left( \frac{k}{ln2} \right)^n 4^{1/k}$$

where  $S$  is number of untimed states,  $n$  is the number of rules enabled concurrently, and  $k$  is maximum timing constraint.

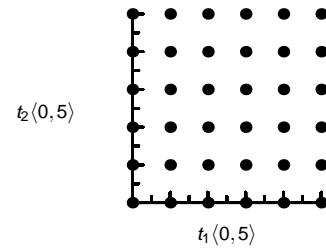
## Discrete-Time

- For timed Petri-nets and TEL structures, all timing requirements are of the form  $\leq$  or  $\geq$ , since timing bounds are inclusive.
- In this case, fractional components are not necessary.
- Only need to track *discrete-time* states.
- Worst-case complexity is now:

$$|S|(k+1)^n$$

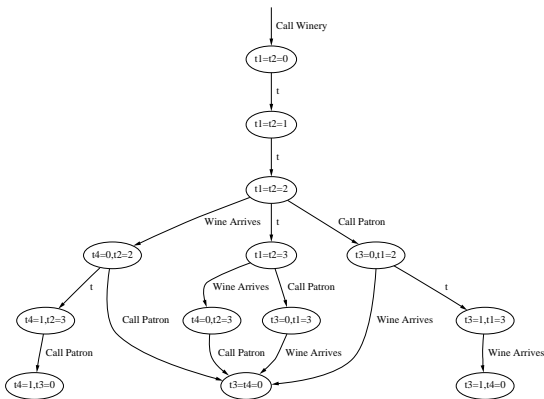
- Reduction by a factor of more than  $n!$ .

## Possible Timed States Using Discrete-Time



36 distinct timed states

## Timed State Space Using Discrete-Time

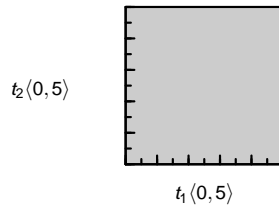


## Timed State Space Explosion Again

- Unfortunately, the discrete-time technique is still exponential in the number of concurrent timers and size of the timing bounds.
- Changing each timing bound of  $[2, 3]$  to  $[19, 31]$  and  $[5, \text{inf}]$  to  $[53, \text{inf}]$ , number of timed states goes from 69 to more than 3000.
- Changing each timing bound of  $[2, 3]$  to  $[191, 311]$  and  $[5, \text{inf}]$  to  $[531, \text{inf}]$ , number of timed states goes to over 300,000!

## Zones

- Another approach is to use convex polygons, called *zones*, to represent equivalence classes of timed states.
- One zone is representing 171 regions or 36 discrete-time states.



## Representing Zones using Linear Inequalities

- Convex polygons can be represented using linear inequalities.
- Introduce a dummy timer  $t_0$  which always takes the value 0.
- For each pair of timers, introduce inequality of the form:

$$t_j - t_i \leq m_{ij}$$

- Example:

$$\begin{array}{l} t_0 - t_0 \leq 0 \\ t_1 - t_0 \leq 5 \\ t_2 - t_0 \leq 5 \\ t_0 - t_1 \leq 0 \\ t_1 - t_1 \leq 0 \end{array} \quad \begin{array}{l} t_2 - t_1 \leq 5 \\ t_0 - t_2 \leq 0 \\ t_1 - t_2 \leq 5 \\ t_2 - t_2 \leq 0 \end{array}$$

## Difference Bound Matrices

- Set of inequalities can be collected into a data structure called a *difference bound matrix* (DBM).
- The difference bound matrix for this example is shown below:

$$\begin{array}{l} t_0 \\ t_1 \\ t_2 \end{array} \begin{array}{l} t_0 \\ t_1 \\ t_2 \end{array} \begin{array}{l} t_2 \\ t_1 \\ t_0 \end{array} \left| \begin{array}{ccc} 0 & 5 & 5 \\ 0 & 0 & 5 \\ 0 & 5 & 0 \end{array} \right.$$

## Recanonicalization

- Many DBMs represent the same zone.
- Need unique DBM representation to determine when a zone has been seen before during the depth first search.
- Each zone has a canonical DBM representation when all entries are minimal.

## Recanonicalization Example

$$\begin{array}{l|ll} & t_0 & t_1 & t_2 \\ t_0 & 0 & 5 & 5 \\ t_1 & 0 & 0 & 7 \\ t_2 & 0 & 5 & 0 \end{array} \quad \begin{array}{l} t_2 - t_1 \leq 7 \\ t_2 - t_0 \leq 5 \\ t_0 - t_1 \leq 0 \end{array}$$

- Add last two equations to get:

$$t_2 - t_1 \leq 5$$

$$\begin{array}{l|ll} & t_0 & t_1 & t_2 \\ t_0 & 0 & 5 & 5 \\ t_1 & 0 & 0 & 5 \\ t_2 & 0 & 5 & 0 \end{array}$$

## DBM as Digraph

- Recanonicalization equivalent to all pairs shortest path problem.
- Create a labeled digraph where:
  - There is a vertex for each timer  $t_i$ ,
  - An arc from  $t_i$  to  $t_j$  for each linear inequality of the form  $t_j - t_i \leq m_{ij}$  when  $i \neq j$ .
  - Each arc is labeled by  $m_{ij}$ .

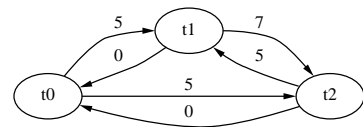
## Floyd's Algorithm

```

recanonicalization(M)
for k = 1 to n
  for i = 1 to n
    for j = 1 to n
      if (mij > mik + mkj) then
        mij = mik + mkj
    
```

## Floyd's Algorithm Example

$$\begin{array}{l|ll} & t_0 & t_1 & t_2 \\ t_0 & 0 & 5 & 5 \\ t_1 & 0 & 0 & 7 \\ t_2 & 0 & 5 & 0 \end{array}$$



## Zone Creation

- After a rule fires:
  - Restrict to reflect minimum firing time.
  - Recononicalize
  - Project out information on rule that fired.
  - Extend matrix with new rows and columns for new rules.
  - Advance time
  - Recononicalize

## Restrict

- Example: firing of a rule  $r_k = \langle e_k, f_k, l_k, u_k \rangle$  where
  - $e_k$  is the enabling event,
  - $f_k$  is the enabled event,
  - $l_k$  is the lower bound of the coresponding timer  $t_k$ , and
  - $u_k$  is the upper bound on the timer.
- Constrain DBM to indicate rule has reached its lower bound.
- $t_0 - t_k \leq -l_k$ , so set  $m_{k0}$  to  $-l_k$ .
- DBM may no longer be maximally tight.
- Recononicalize DBM using Floyd's algorithm.

## Project

- Remove the row and column corresponding to  $t_k$ .
- If rule firing causes an event, new rules may be enabled.
- For newly enabled rules, introduce a new timer  $t_i$  with a row and column in the DBM.
- Initialize  $m_{j0}$  and  $m_{0i}$  to 0.
- Initialize each  $m_{ij}$  to  $m_{0j}$ .
- Initialize each  $m_{ij}$  to  $m_{j0}$ .

## Advance Time

- Set all timers to their upper bound (i.e.,  $m_{0j} = u_j$ ).
- Recanonicalize the DBM using Floyd's algorithm.

## Update Zone

```

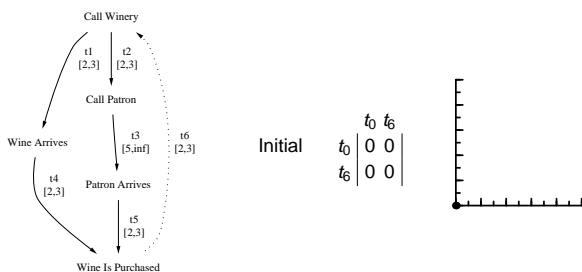
update_zone(M, r_j, event_fired, R_en, R_new)
  if m_j0 > -l_j then m_j0 = -l_j
  recanonicalize(M)
  project(M, r_j)
  if (event_fired) then
    foreach r_i in R_new
      m_i0 = m_0i = 0
    foreach r_k in R_new
      m_kk = m_k0 = 0
    foreach r_k in (R_en - R_new)
      m_kk = m_0k
      m_ki = m_k0
    foreach r_i in R_en
      m_0i = u_i
  recanonicalize(M)
  normalize(M, R_en)
  
```

## Normalize

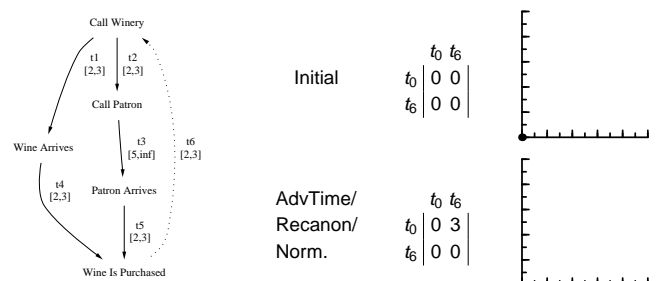
```

normalize(M, R_en)
  foreach r_i in R_en
    if (m_i0 < -premax(r_i)) then
      foreach r_j in R_en
        m_ij = m_ij - (m_i0 + premax(r_i))
        m_ji = m_ji + (m_i0 + premax(r_i))
    foreach r_i in R_en
      if (m_0i > premax(r_i)) then
        m_0i = max_j(min(m_0j, premax(r_j)) - m_ij)
  recanonicalize(M)
  
```

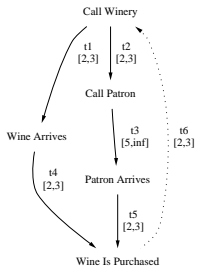
## Initial Zone



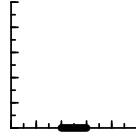
## Initial Zone



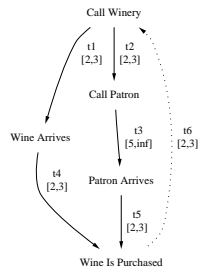
### Zone After Winery Called



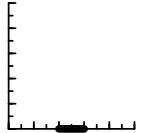
$$\text{Restrict/Recanon} \quad \begin{matrix} t_0 & t_6 \\ t_0 & 0 & 3 \\ t_6 & -2 & 0 \end{matrix}$$



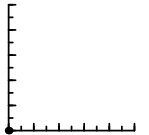
### Zone After Winery Called



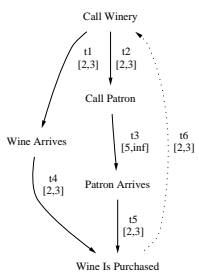
$$\text{Restrict/Recanon} \quad \begin{matrix} t_0 & t_6 \\ t_0 & 0 & 3 \\ t_6 & -2 & 0 \end{matrix}$$



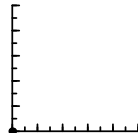
$$\text{Project} \quad \begin{matrix} t_0 \\ t_0 & 0 \end{matrix}$$



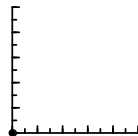
### Zone After Winery Called



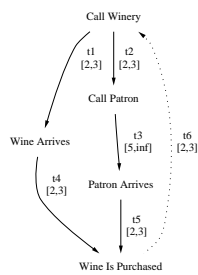
$$\text{Project} \quad \begin{matrix} t_0 \\ t_0 & 0 \end{matrix}$$



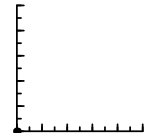
$$\text{Extend} \quad \begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 0 & 0 \\ t_1 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 0 \end{matrix}$$



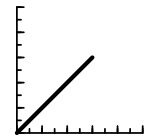
### Zone After Winery Called



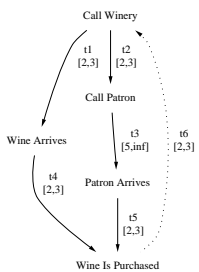
$$\text{Extend} \quad \begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 0 & 0 \\ t_1 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 0 \end{matrix}$$



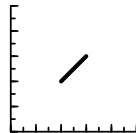
$$\text{AdvTime/Recanon/ Norm.} \quad \begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 0 \end{matrix}$$



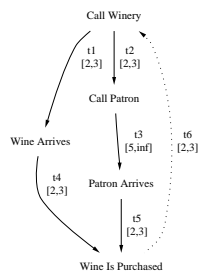
### Zone After Wine Arrives



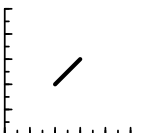
$$\text{Restrict} \quad \begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & -2 & 0 & 0 \\ t_2 & 0 & 0 & 0 \end{matrix}$$



### Zone After Wine Arrives



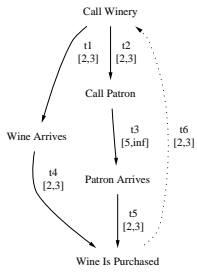
$$\text{Restrict} \quad \begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & -2 & 0 & 0 \\ t_2 & 0 & 0 & 0 \end{matrix}$$



$$\text{Recanon} \quad \begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & -2 & 0 & 0 \\ t_2 & -2 & 0 & 0 \end{matrix}$$

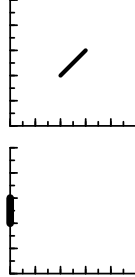


### Zone After Wine Arrives

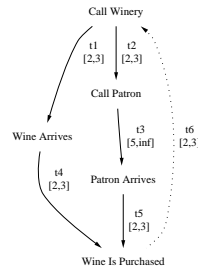


$$\text{Reconon} \begin{array}{c} t_0 \quad t_1 \quad t_2 \\ t_0 \quad 0 \quad 3 \quad 3 \\ t_1 \quad -2 \quad 0 \quad 0 \\ t_2 \quad -2 \quad 0 \quad 0 \end{array}$$

$$\text{Project} \begin{array}{c} t_0 \quad t_2 \\ t_0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad 0 \end{array}$$

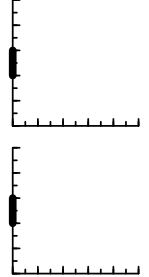


### Zone After Wine Arrives

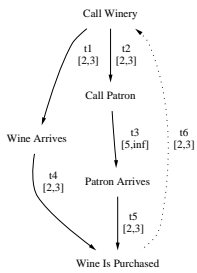


$$\text{Project} \begin{array}{c} t_0 \quad t_2 \\ t_0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad 0 \end{array}$$

$$\text{Extend} \begin{array}{c} t_0 \quad t_4 \quad t_2 \\ t_0 \quad 0 \quad 0 \quad 3 \\ t_4 \quad 0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad -2 \quad 0 \end{array}$$

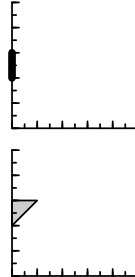


### Zone After Wine Arrives

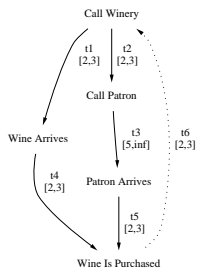


$$\text{Extend} \begin{array}{c} t_0 \quad t_4 \quad t_2 \\ t_0 \quad 0 \quad 0 \quad 3 \\ t_4 \quad 0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad -2 \quad 0 \end{array}$$

$$\text{AdvTime} \begin{array}{c} t_0 \quad t_4 \quad t_2 \\ t_0 \quad 0 \quad 3 \quad 3 \\ t_4 \quad 0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad -2 \quad 0 \end{array}$$

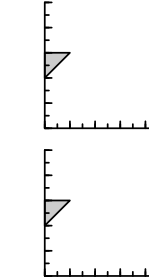


### Zone After Wine Arrives

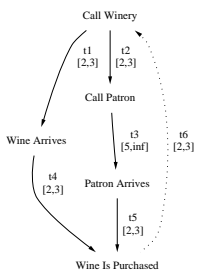


$$\text{AdvTime} \begin{array}{c} t_0 \quad t_4 \quad t_2 \\ t_0 \quad 0 \quad 3 \quad 3 \\ t_4 \quad 0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad -2 \quad 0 \end{array}$$

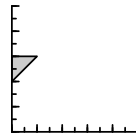
$$\text{Reconon/ Norm.} \begin{array}{c} t_0 \quad t_4 \quad t_2 \\ t_0 \quad 0 \quad 1 \quad 3 \\ t_4 \quad 0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad -2 \quad 0 \end{array}$$



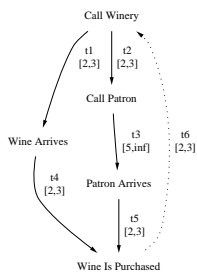
### Zone After Wine Arrives and Patron is Called



$$\text{Restrict/ Reconon} \begin{array}{c} t_0 \quad t_4 \quad t_2 \\ t_0 \quad 0 \quad 1 \quad 3 \\ t_4 \quad 0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad -2 \quad 0 \end{array}$$

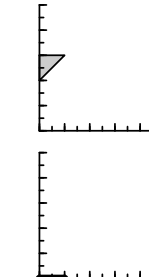


### Zone After Wine Arrives and Patron is Called

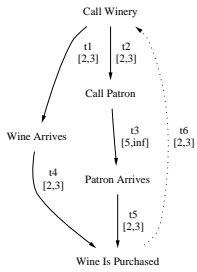


$$\text{Restrict/ Reconon} \begin{array}{c} t_0 \quad t_4 \quad t_2 \\ t_0 \quad 0 \quad 1 \quad 3 \\ t_4 \quad 0 \quad 0 \quad 3 \\ t_2 \quad -2 \quad -2 \quad 0 \end{array}$$

$$\text{Project} \begin{array}{c} t_0 \quad t_4 \\ t_0 \quad 0 \quad 1 \\ t_4 \quad 0 \quad 0 \end{array}$$



### Zone After Wine Arrives and Patron is Called

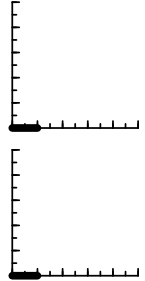


Project

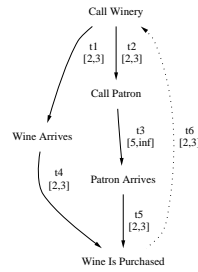
$$\begin{matrix} & t_0 & t_4 \\ t_0 & 0 & 1 \\ t_4 & 0 & 0 \end{matrix}$$

Extend

$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 1 & 0 \\ t_4 & 0 & 0 & 0 \\ t_3 & 0 & 1 & 0 \end{matrix}$$



### Zone After Wine Arrives and Patron is Called

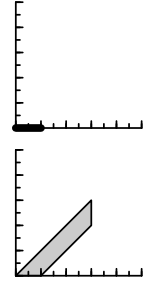


Extend

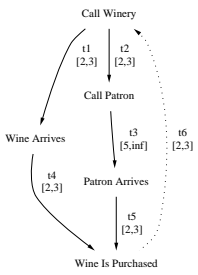
$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 1 & 0 \\ t_4 & 0 & 0 & 0 \\ t_3 & 0 & 1 & 0 \end{matrix}$$

AdvTime

$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & \infty \\ t_4 & 0 & 0 & 0 \\ t_3 & 0 & 1 & 0 \end{matrix}$$



### Zone After Wine Arrives and Patron is Called

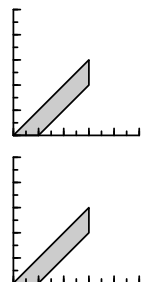


AdvTime

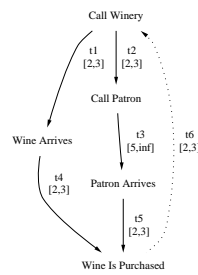
$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & \infty \\ t_4 & 0 & 0 & 0 \\ t_3 & 0 & 1 & 0 \end{matrix}$$

Reconon/  
Norm.

$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 3 \\ t_4 & 0 & 0 & 0 \\ t_3 & 0 & 1 & 0 \end{matrix}$$

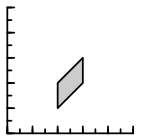


### Zone After Rule Expires

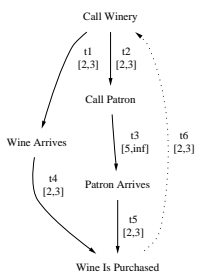


Restrict

$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 3 \\ t_4 & -2 & 0 & 0 \\ t_3 & 0 & 1 & 0 \end{matrix}$$



### Zone After Rule Expires

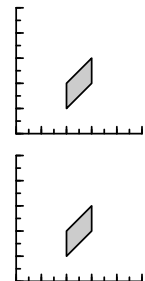


Restrict

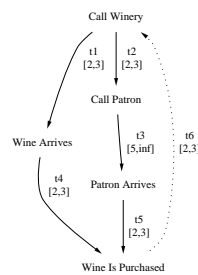
$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 3 \\ t_4 & -2 & 0 & 0 \\ t_3 & 0 & 1 & 0 \end{matrix}$$

Reconon

$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 3 \\ t_4 & -2 & 0 & 0 \\ t_3 & -1 & 1 & 0 \end{matrix}$$



### Zone After Rule Expires

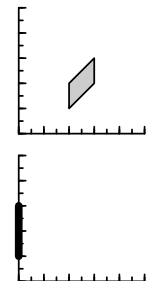


Reconon

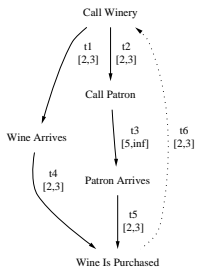
$$\begin{matrix} & t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 3 \\ t_4 & -2 & 0 & 0 \\ t_3 & -1 & 1 & 0 \end{matrix}$$

Project

$$\begin{matrix} & t_0 & t_3 \\ t_0 & 0 & 3 \\ t_3 & -1 & 0 \end{matrix}$$

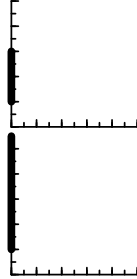


### Zone After Rule Expires

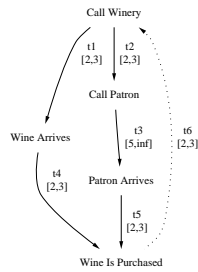


$$\text{Project} \quad \begin{array}{c|cc} & t_0 & t_3 \\ t_0 & 0 & 3 \\ t_3 & -1 & 0 \end{array}$$

$$\text{AdvTime/Reconon} \quad \begin{array}{c|cc} & t_0 & t_3 \\ t_0 & 0 & \infty \\ t_3 & -1 & 0 \end{array}$$

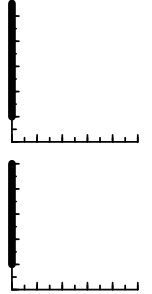


### Zone After Rule Expires

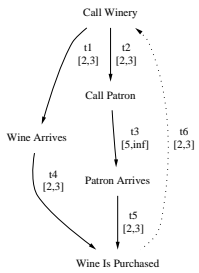


$$\text{AdvTime/Reconon} \quad \begin{array}{c|cc} & t_0 & t_3 \\ t_0 & 0 & \infty \\ t_3 & -1 & 0 \end{array}$$

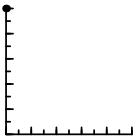
$$\text{Norm.} \quad \begin{array}{c|cc} & t_0 & t_3 \\ t_0 & 0 & 5 \\ t_3 & -1 & 0 \end{array}$$



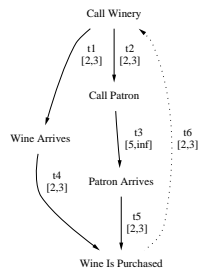
### Zone After Patron Arrives



$$\text{Restrict/Reconon} \quad \begin{array}{c|cc} & t_0 & t_3 \\ t_0 & 0 & 5 \\ t_3 & -5 & 0 \end{array}$$

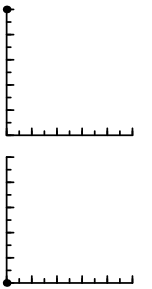


### Zone After Patron Arrives

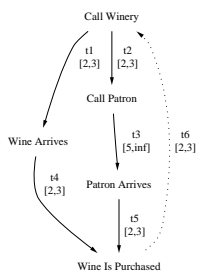


$$\text{Restrict/Reconon} \quad \begin{array}{c|cc} & t_0 & t_3 \\ t_0 & 0 & 5 \\ t_3 & -5 & 0 \end{array}$$

$$\text{Project} \quad \begin{array}{c|c} & t_0 \\ t_0 & 0 \end{array}$$

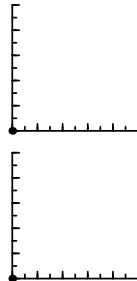


### Zone After Patron Arrives

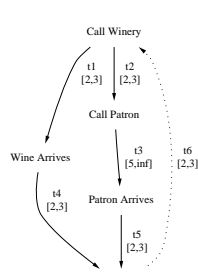


$$\text{Project} \quad \begin{array}{c|c} & t_0 \\ t_0 & 0 \end{array}$$

$$\text{Extend} \quad \begin{array}{c|ccc} & t_0 & t_5 & \\ t_0 & 0 & 0 & 0 \\ t_5 & 0 & 0 & 0 \end{array}$$

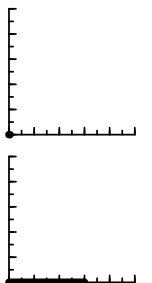


### Zone After Patron Arrives

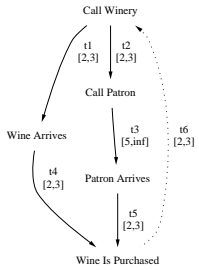


$$\text{Extend} \quad \begin{array}{c|ccc} & t_0 & t_5 & \\ t_0 & 0 & 0 & 0 \\ t_5 & 0 & 0 & 0 \end{array}$$

$$\begin{array}{l} \text{AdvTime/} \\ \text{Reconon/} \\ \text{Norm.} \end{array} \quad \begin{array}{c|ccc} & t_0 & t_5 & \\ t_0 & 0 & 3 & \\ t_5 & 0 & 0 & \end{array}$$

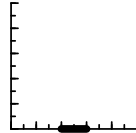


### Zone After Wine is Purchased

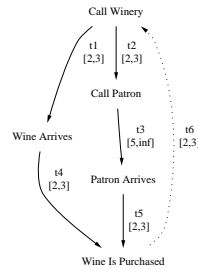


Restrict/  
Recanon

$$\begin{matrix} t_0 & t_5 \\ t_0 & 0 & 3 \\ t_5 & -2 & 0 \end{matrix}$$

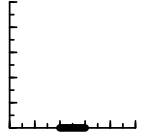


### Zone After Wine is Purchased



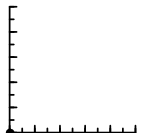
Restrict/  
Recanon

$$\begin{matrix} t_0 & t_5 \\ t_0 & 0 & 3 \\ t_5 & -2 & 0 \end{matrix}$$

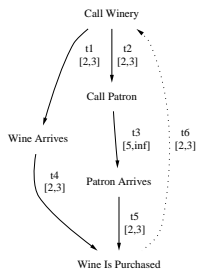


Project

$$\begin{matrix} t_0 \\ t_0 & 0 \end{matrix}$$

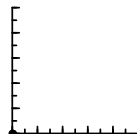


### Zone After Wine is Purchased



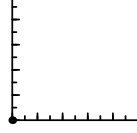
Project

$$\begin{matrix} t_0 \\ t_0 & 0 \end{matrix}$$

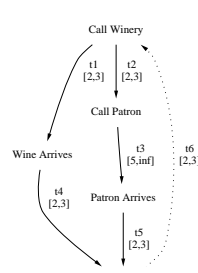


Extend

$$\begin{matrix} t_0 & t_6 \\ t_0 & 0 & 0 \\ t_6 & 0 & 0 \end{matrix}$$

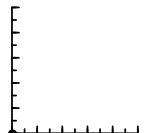


### Zone After Wine is Purchased



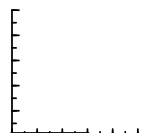
Extend

$$\begin{matrix} t_0 & t_6 \\ t_0 & 0 & 0 \\ t_6 & 0 & 0 \end{matrix}$$

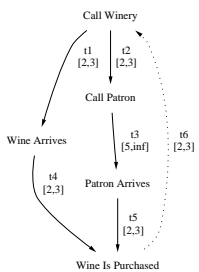


AdvTime/  
Recanon/  
Norm.

$$\begin{matrix} t_0 & t_6 \\ t_0 & 0 & 3 \\ t_6 & 0 & 0 \end{matrix}$$

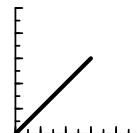


### Zone After Patron is Called

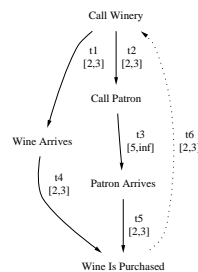


Initial

$$\begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 0 \end{matrix}$$

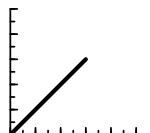


### Zone After Patron is Called



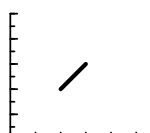
Initial

$$\begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 0 \end{matrix}$$

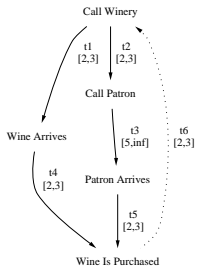


Restrict

$$\begin{matrix} t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & 0 & 0 & 0 \\ t_2 & -2 & 0 & 0 \end{matrix}$$



### Zone After Patron is Called

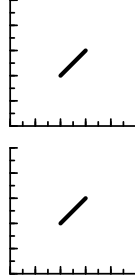


Restrict

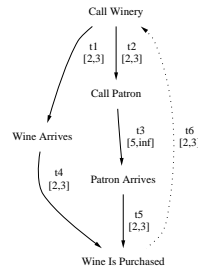
$$\begin{matrix} & t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & 0 & 0 & 0 \\ t_2 & -2 & 0 & 0 \end{matrix}$$

Recanon

$$\begin{matrix} & t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & -2 & 0 & 0 \\ t_2 & -2 & 0 & 0 \end{matrix}$$



### Zone After Patron is Called

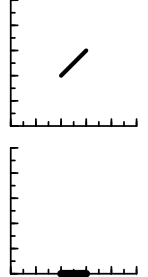


Recanon

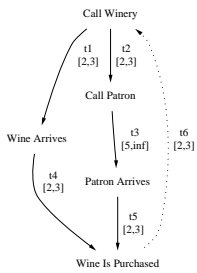
$$\begin{matrix} & t_0 & t_1 & t_2 \\ t_0 & 0 & 3 & 3 \\ t_1 & -2 & 0 & 0 \\ t_2 & -2 & 0 & 0 \end{matrix}$$

Project

$$\begin{matrix} & t_0 & t_1 \\ t_0 & 0 & 3 \\ t_1 & -2 & 0 \end{matrix}$$



### Zone After Patron is Called

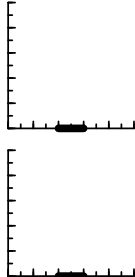


Project

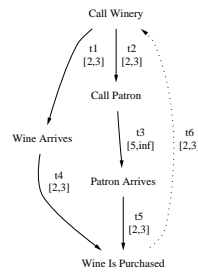
$$\begin{matrix} & t_0 & t_1 \\ t_0 & 0 & 3 \\ t_1 & -2 & 0 \end{matrix}$$

Extend

$$\begin{matrix} & t_0 & t_1 & t_3 \\ t_0 & 0 & 3 & 0 \\ t_1 & -2 & 0 & -2 \\ t_3 & 0 & 3 & 0 \end{matrix}$$



### Zone After Patron is Called

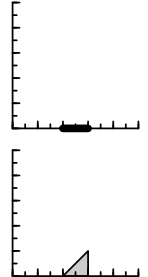


Extend

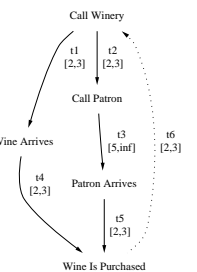
$$\begin{matrix} & t_0 & t_1 & t_3 \\ t_0 & 0 & 3 & 0 \\ t_1 & -2 & 0 & -2 \\ t_3 & 0 & 3 & 0 \end{matrix}$$

AdvTime

$$\begin{matrix} & t_0 & t_1 & t_3 \\ t_0 & 0 & 3 & \infty \\ t_1 & -2 & 0 & -2 \\ t_3 & 0 & 3 & 0 \end{matrix}$$



### Zone After Patron is Called

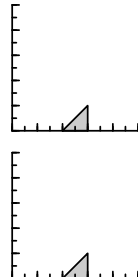


AdvTime

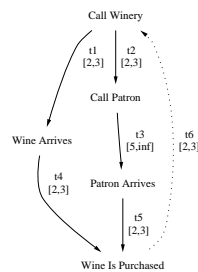
$$\begin{matrix} & t_0 & t_1 & t_3 \\ t_0 & 0 & 3 & \infty \\ t_1 & -2 & 0 & -2 \\ t_3 & 0 & 3 & 0 \end{matrix}$$

Recanon/  
Norm.

$$\begin{matrix} & t_0 & t_1 & t_3 \\ t_0 & 0 & 3 & 1 \\ t_1 & -2 & 0 & -2 \\ t_3 & 0 & 3 & 0 \end{matrix}$$

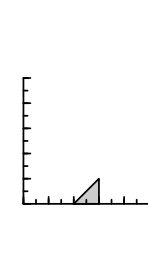


### Zone After Patron is Called and Wine Arrives

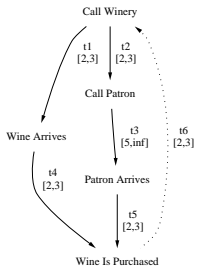


Restrict/  
Recanon

$$\begin{matrix} & t_0 & t_1 & t_3 \\ t_0 & 0 & 3 & 1 \\ t_1 & -2 & 0 & -2 \\ t_3 & 0 & 3 & 0 \end{matrix}$$

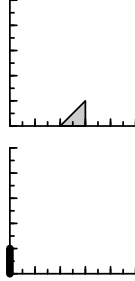


### Zone After Patron is Called and Wine Arrives

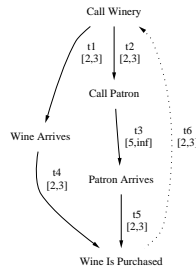


$$\text{Restrict/Recanon} \quad \begin{matrix} t_0 & t_1 & t_3 \\ t_0 & 0 & 3 & 1 \\ t_1 & -2 & 0 & -2 \\ t_3 & 0 & 3 & 0 \end{matrix}$$

$$\text{Project} \quad \begin{matrix} t_0 & t_3 \\ t_0 & 0 & 1 \\ t_3 & 0 & 0 \end{matrix}$$

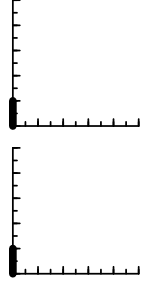


### Zone After Patron is Called and Wine Arrives

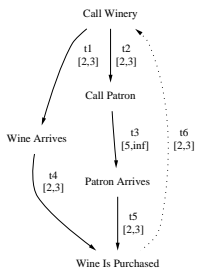


$$\text{Project} \quad \begin{matrix} t_0 & t_3 \\ t_0 & 0 & 1 \\ t_3 & 0 & 0 \end{matrix}$$

$$\text{Extend} \quad \begin{matrix} t_0 & t_4 & t_3 \\ t_0 & 0 & 0 & 1 \\ t_4 & 0 & 0 & 1 \\ t_3 & 0 & 0 & 0 \end{matrix}$$

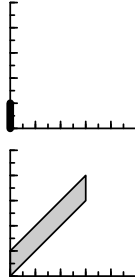


### Zone After Patron is Called and Wine Arrives

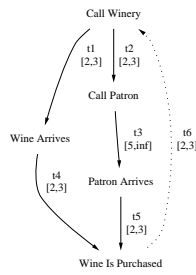


$$\text{Extend} \quad \begin{matrix} t_0 & t_4 & t_3 \\ t_0 & 0 & 0 & 1 \\ t_4 & 0 & 0 & 1 \\ t_3 & 0 & 0 & 0 \end{matrix}$$

$$\text{AdvTime} \quad \begin{matrix} t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & \infty \\ t_4 & 0 & 0 & 1 \\ t_3 & 0 & 0 & 0 \end{matrix}$$

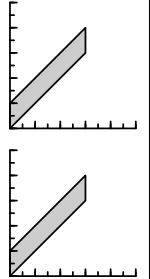


### Zone After Patron is Called and Wine Arrives

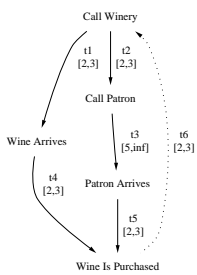


$$\text{AdvTime} \quad \begin{matrix} t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & \infty \\ t_4 & 0 & 0 & 1 \\ t_3 & 0 & 0 & 0 \end{matrix}$$

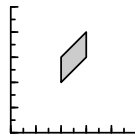
$$\text{Recanon/Norm.} \quad \begin{matrix} t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 4 \\ t_4 & 0 & 0 & 1 \\ t_3 & 0 & 0 & 0 \end{matrix}$$



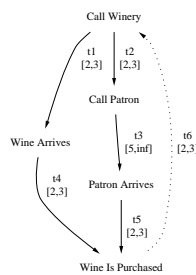
### Zone After Rule Expires



$$\text{Restrict} \quad \begin{matrix} t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 4 \\ t_4 & -2 & 0 & 1 \\ t_3 & 0 & 0 & 0 \end{matrix}$$

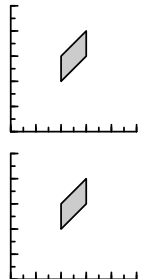


### Zone After Rule Expires

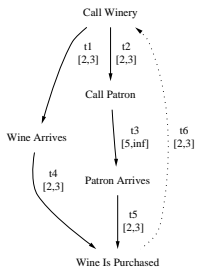


$$\text{Restrict} \quad \begin{matrix} t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 4 \\ t_4 & -2 & 0 & 1 \\ t_3 & 0 & 0 & 0 \end{matrix}$$

$$\text{Recanon} \quad \begin{matrix} t_0 & t_4 & t_3 \\ t_0 & 0 & 3 & 4 \\ t_4 & -2 & 0 & 1 \\ t_3 & -2 & 0 & 0 \end{matrix}$$

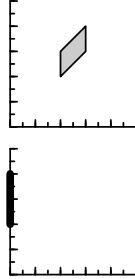


### Zone After Rule Expires

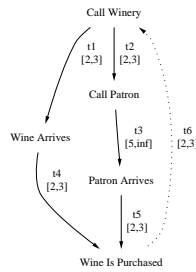


$$\text{Recanon} \begin{array}{c|ccc} & t_0 & t_4 & t_3 \\ \hline t_0 & 0 & 3 & 4 \\ t_4 & -2 & 0 & 1 \\ t_3 & -2 & 0 & 0 \end{array}$$

$$\text{Project} \begin{array}{c|cc} & t_0 & t_3 \\ \hline t_0 & 0 & 4 \\ t_3 & -2 & 0 \end{array}$$

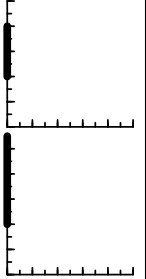


### Zone After Rule Expires

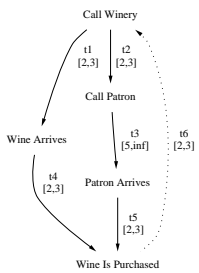


$$\text{Project} \begin{array}{c|cc} & t_0 & t_3 \\ \hline t_0 & 0 & 4 \\ t_3 & -2 & 0 \end{array}$$

$$\text{AdvTime / Recanon} \begin{array}{c|cc} & t_0 & t_3 \\ \hline t_0 & 0 & \infty \\ t_3 & -2 & 0 \end{array}$$

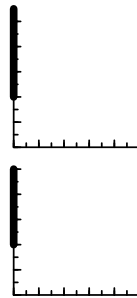


### Zone After Rule Expires

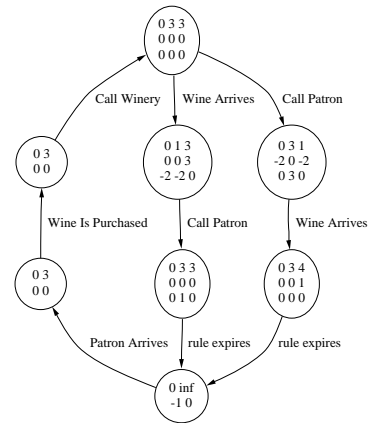


$$\text{AdvTime / Recanon} \begin{array}{c|cc} & t_0 & t_3 \\ \hline t_0 & 0 & \infty \\ t_3 & -2 & 0 \end{array}$$

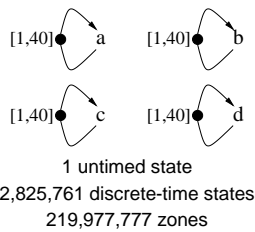
$$\text{Normalize} \begin{array}{c|cc} & t_0 & t_3 \\ \hline t_0 & 0 & 5 \\ t_3 & -2 & 0 \end{array}$$



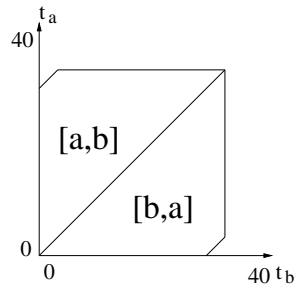
### Timed State Space using Zones



### Adverse Example



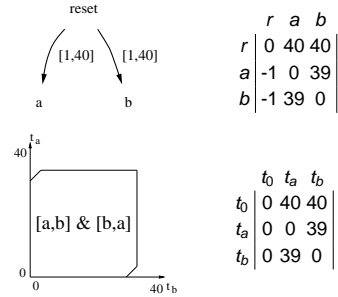
### Order versus Causality



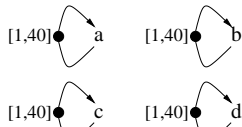
## POSET Timing

- Using linear traces introduces fake orderings.
- Need to separate concurrency from causality.
- Find zones on POSETs rather than linear traces.
- Represent POSETs using graph/matrix.

## POSET Graph/Matrix/Zone



## POSET Timing

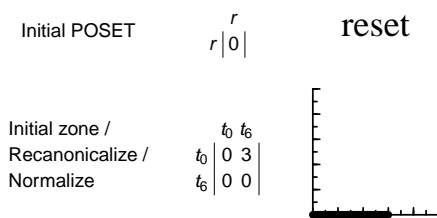


1 untimed state  
 2,825,761 discrete-time states  
 219,977,777 zones  
 1 zone found using POSET timing

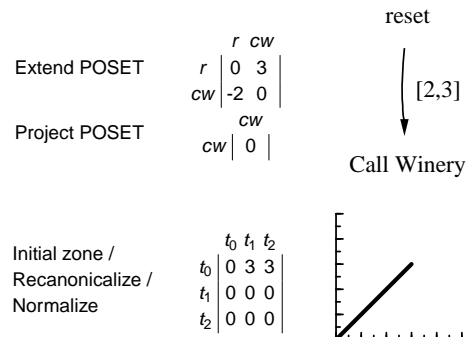
## Creating New Zones

- If event occurs, update POSET matrix and create zone:
  - Set minimums to 0 (i.e.,  $m_{j0} = 0$ ).
  - Set maximums to the upper bound (i.e.,  $m_{0j} = u_j$ ).
  - Copy relevant time separations from POSET matrix to zone (i.e.,  $m_{ij} = p_{ij}$ ).
  - Recanonicalize.
- Otherwise, project out timer corresponding to rule that fired.

## Initial Zone using POSETs



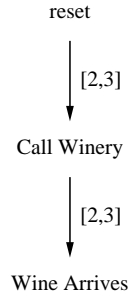
## Zone after the Winery is Called using POSETs



### POSET after the Wine Arrives

Extend POSET

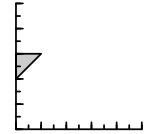
$$\begin{array}{c|cc} & cw & wa \\ \hline cw & 0 & 3 \\ wa & -2 & 0 \end{array}$$



### Zone after the Wine Arrives using POSETs

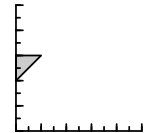
Initial zone

$$\begin{array}{c|ccc} & t_0 & t_4 & t_2 \\ \hline t_0 & 0 & 3 & 3 \\ t_4 & 0 & 0 & 3 \\ t_2 & 0 & -2 & 0 \end{array}$$



Recanonicalize / Normalize

$$\begin{array}{c|ccc} & t_0 & t_4 & t_2 \\ \hline t_0 & 0 & 1 & 3 \\ t_4 & 0 & 0 & 3 \\ t_2 & -2 & -2 & 0 \end{array}$$



### POSET after the Patron is Called

Extend POSET

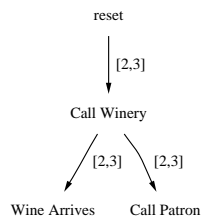
$$\begin{array}{c|ccc} & cw & wa & cp \\ \hline cw & 0 & 3 & 3 \\ wa & -2 & 0 & \infty \\ cp & -2 & \infty & 0 \end{array}$$

Recanonicalize

$$\begin{array}{c|ccc} & cw & wa & cp \\ \hline cw & 0 & 3 & 3 \\ wa & -2 & 0 & 1 \\ cp & -2 & 1 & 0 \end{array}$$

Project POSET

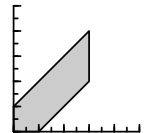
$$\begin{array}{c|cc} & wa & cp \\ \hline wa & 0 & 1 \\ cp & 1 & 0 \end{array}$$



### Zone after the Patron is Called using POSETs

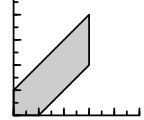
Initial zone

$$\begin{array}{c|ccc} & t_0 & t_4 & t_3 \\ \hline t_0 & 0 & 3 & \infty \\ t_4 & 0 & 0 & 1 \\ t_3 & 0 & 1 & 0 \end{array}$$



Recanonicalize / Normalize

$$\begin{array}{c|ccc} & t_0 & t_4 & t_3 \\ \hline t_0 & 0 & 3 & 4 \\ t_4 & 0 & 0 & 1 \\ t_3 & 0 & 1 & 0 \end{array}$$



### Zone after the Rule Expires using POSETs

Project zone

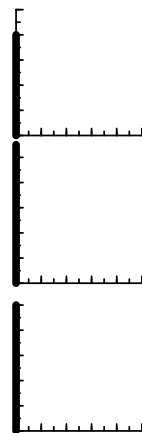
$$\begin{array}{c|cc} & t_0 & t_3 \\ \hline t_0 & 0 & 4 \\ t_3 & 0 & 0 \end{array}$$

Advance time / Recanonicalize

$$\begin{array}{c|cc} & t_0 & t_3 \\ \hline t_0 & 0 & \infty \\ t_3 & 0 & 0 \end{array}$$

Normalize

$$\begin{array}{c|cc} & t_0 & t_3 \\ \hline t_0 & 0 & 5 \\ t_3 & 0 & 0 \end{array}$$



### POSET after the Patron Arrives

Extend POSET

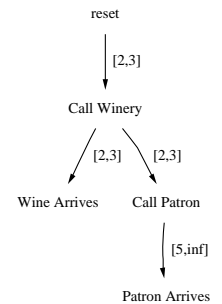
$$\begin{array}{c|ccc} & wa & cp & pa \\ \hline wa & 0 & 1 & \infty \\ cp & 1 & 0 & \infty \\ pa & \infty & -5 & 0 \end{array}$$

Recanonicalize

$$\begin{array}{c|ccc} & wa & cp & pa \\ \hline wa & 0 & 1 & \infty \\ cp & 1 & 0 & \infty \\ pa & -4 & -5 & 0 \end{array}$$

Project POSET

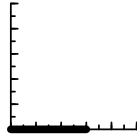
$$\begin{array}{c|c} & pa \\ \hline pa & 0 \end{array}$$



### Zone after the Patron Arrives using POSETs

Initial zone /  
 Recanonicalize /  
 Normalize

$$\begin{matrix} & t_0 & t_5 \\ t_0 & | & 0 & 3 \\ t_5 & | & 0 & 0 \end{matrix}$$

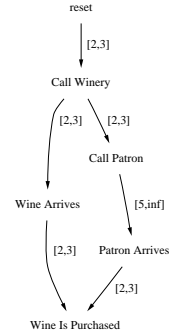


### POSET after the Wine is Purchased

Extend POSET

$$\begin{matrix} & pa & wp \\ pa & | & 0 & 3 \\ wp & | & -2 & 0 \end{matrix}$$

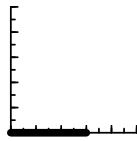
Project POSET

$$\begin{matrix} & wp \\ wp & | & 0 \end{matrix}$$


### Zone after the Wine is Purchased using POSETs

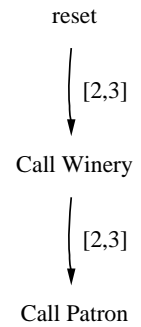
Initial zone /  
 Recanonicalize /  
 Normalize

$$\begin{matrix} & t_0 & t_6 \\ t_0 & | & 0 & 3 \\ t_6 & | & 0 & 0 \end{matrix}$$



### POSET after the Patron is Called

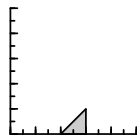
Extend POSET

$$\begin{matrix} & cp & cw \\ cp & | & 0 & -2 \\ cw & | & 3 & 0 \end{matrix}$$


### Zone after the Patron is Called using POSETs

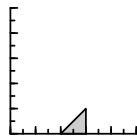
Initial zone

$$\begin{matrix} & t_0 & t_1 & t_3 \\ t_0 & | & 0 & 3 & \infty \\ t_1 & | & 0 & 0 & -2 \\ t_3 & | & 0 & 3 & 0 \end{matrix}$$



Recanonicalize /  
 Normalize

$$\begin{matrix} & t_0 & t_1 & t_3 \\ t_0 & | & 0 & 3 & 1 \\ t_1 & | & -2 & 0 & -2 \\ t_3 & | & 0 & 3 & 0 \end{matrix}$$



### POSET after the Wine Arrives

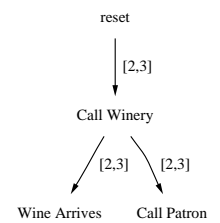
Extend POSET

$$\begin{matrix} & wa & cp & cw \\ wa & | & 0 & \infty & -2 \\ cp & | & \infty & 0 & -2 \\ cw & | & 3 & 3 & 0 \end{matrix}$$

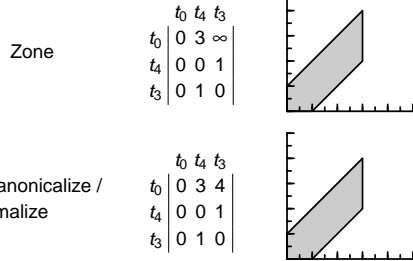
Recanonicalize

$$\begin{matrix} & wa & cp & cw \\ wa & | & 0 & 1 & -2 \\ cp & | & 1 & 0 & -2 \\ cw & | & 3 & 3 & 0 \end{matrix}$$

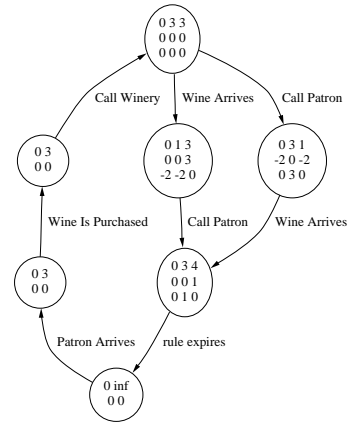
Project POSET

$$\begin{matrix} & wa & cp \\ wa & | & 0 & 1 \\ cp & | & 1 & 0 \end{matrix}$$


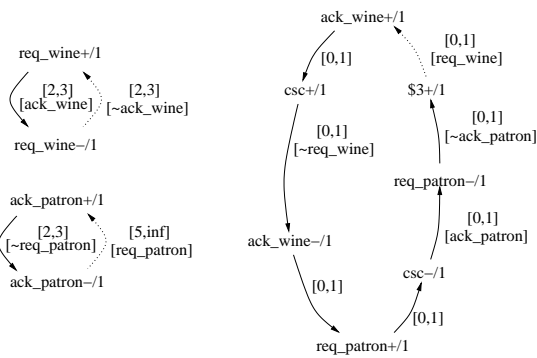
### Zone after the Wine Arrives using POSETs



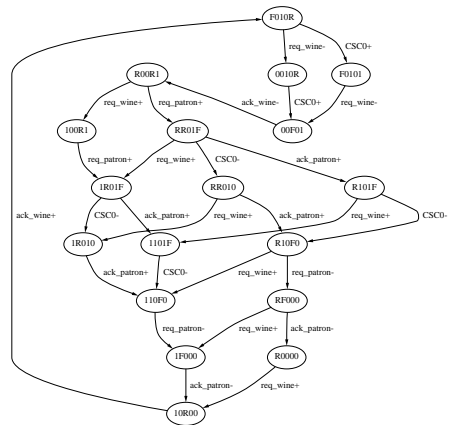
### Timed State Space using POSETs



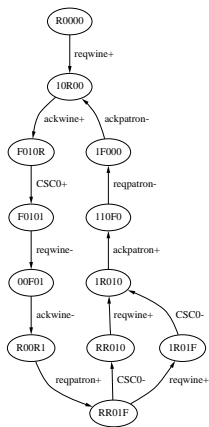
### Wine Shop Example: TEL Structure



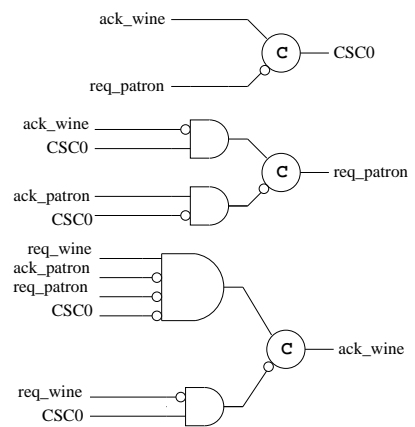
### Wine Shop Example: Untimed State Graph



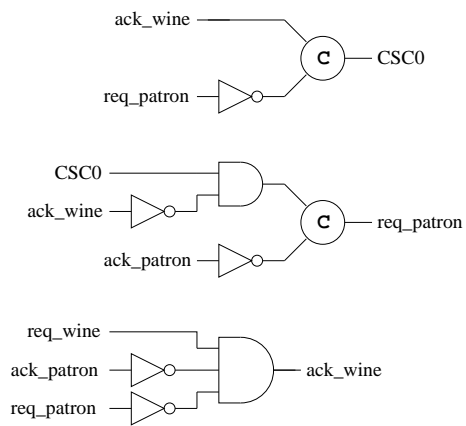
### Wine Shop Example: Reduced State Graph



### Wine Shop Example: Speed-Independent Circuit



## Wine Shop Example: Timed Circuit



## Summary

- Regions
- Discrete-time states
- Zones
- Zones + POSETs
- Timed circuits