Modeling Timed Systems

Alwyn E. Goodloe

- We give an overview of several papers produced by j. Sifakis' group at VERIMAG in France.

- They are working on formal techniques that hope to give designers of real-time distributed systems tools for constructing correct systems in a modular fashion.

- This goal is made difficult because many desirable properties are stated about the global system. What an engineer wants are properties about components that imply the global system properties when the system is composed. They also want to be able to build the system via a process of stepwise refinement and know that that at each stage they are moving towards a correct system.

- In this talk we will present work on Liveness and Scheduling.

Recall that it is standard practice to model systems using automata. (That's why we take Gallier's 511.) Traditional automata don't consider time.

An untimed transition system is:

- S - set of controlled states.
- A - Set of actions.
- transition - $\longrightarrow \subseteq (S \times A \times S)$

In order to model timed systems, we augment the untimed system as follows:

- Let $X = \{x_1, \ldots, x_m\}$ with $x_i \in \mathbf{R}_+$ be a set of clocks.

- A fn that maps untimd to timed systems as follows:

$$h(s, a, s') = (s, a, g, \tau, r, s')$$

Where

- g - is a guard given as a constraint. An inequality on timers and constants.

- $r \subseteq X$ are a set of timers to be rest to 0.

- $\tau \in \{\lambda, \delta, \epsilon\}$ Indicating the urgency of the action. More on this later.

Constraint C is defined as

$$C ::= C\#d | x - y\#d | C \wedge C | \neg C$$

where $x, y \in X$ and $d \in \mathbf{Z}$ and $\#$ is $<$ or $\leq$ In other words, a constraint is an inequality on timers and constants.

Semantics

Transition Graph - $(\mathcal{V}, \mathcal{E})$ Defined from a transition system as follows:

- Vertices - $\mathcal{V} = S \times \mathbf{R}_+$ So States of timed systems are pairs (s,x).
- Edges - $\mathcal{E} \subseteq \mathcal{V} \times (A \times \mathbf{R}_+) \times \mathcal{V}$.

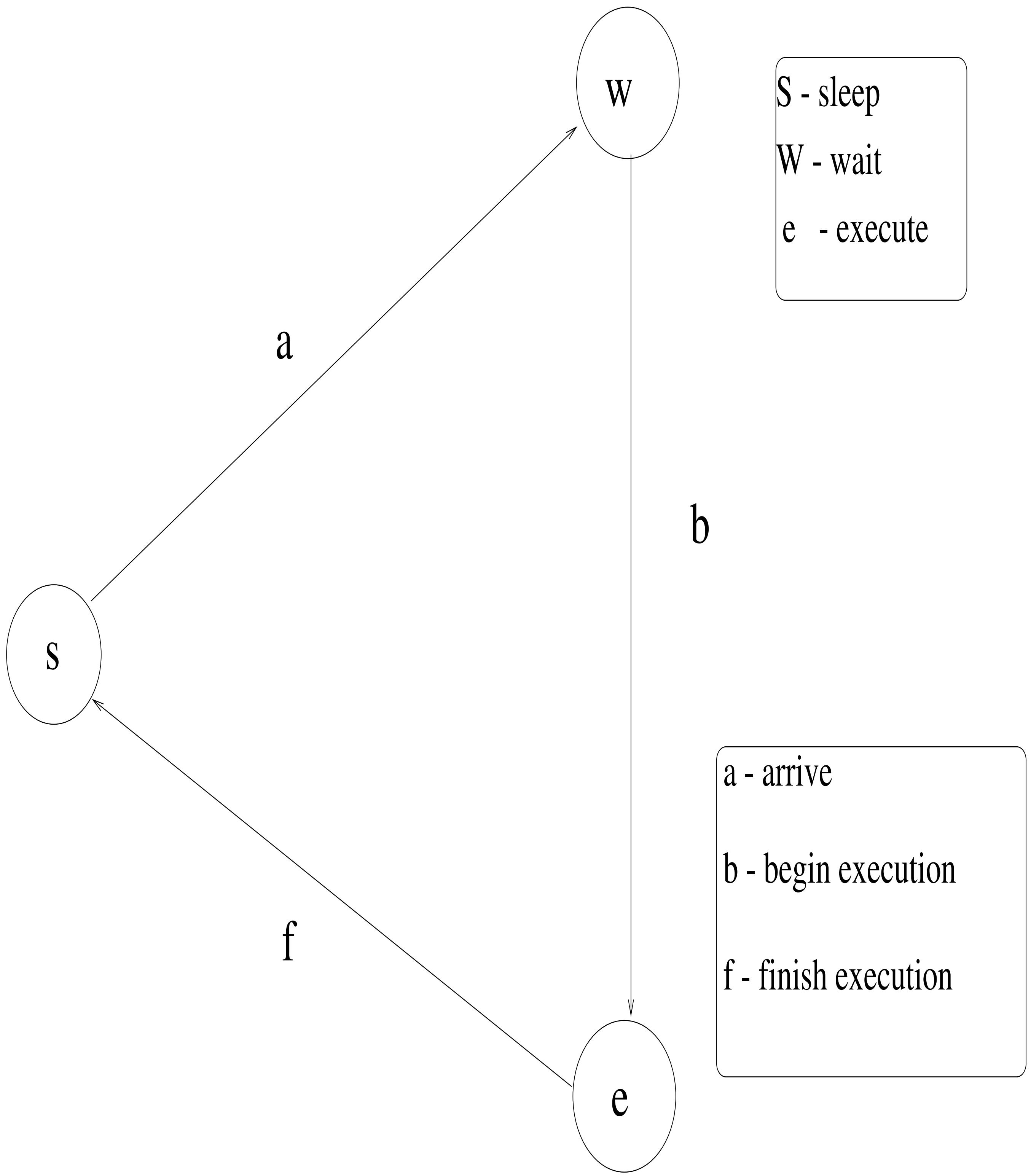
Think of a run of this system as alternating time progress and actions.

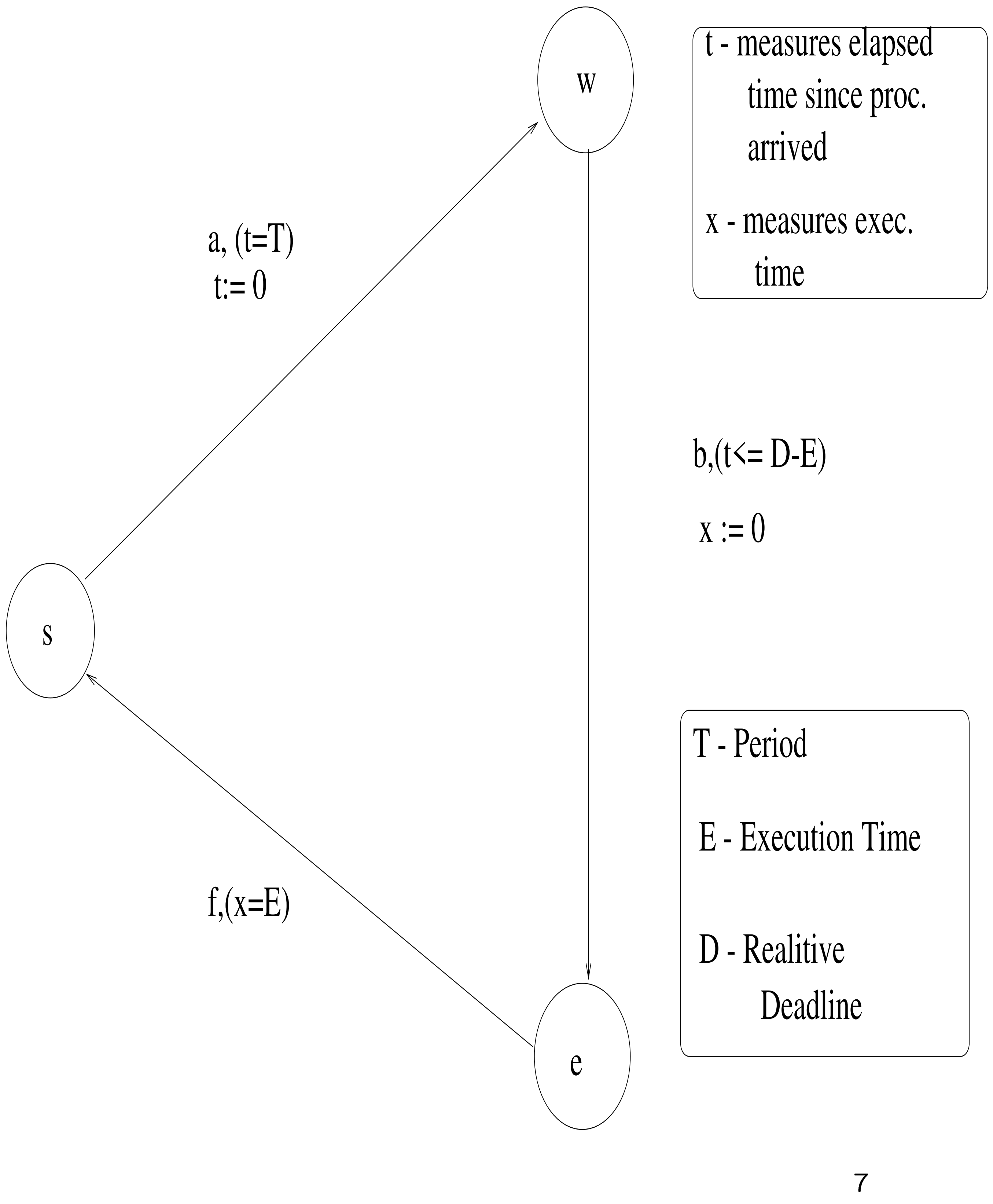
$(s_0, v_0) \xrightarrow{t_0} (s_0, v_0 + t_0) \xrightarrow{a_1} \ldots \xrightarrow{t_i} (s_i, v_i + t_i) \xrightarrow{a_i} \ldots$

note:

$$((s, x), a_j, (s_j, x[r])) \in \mathcal{E} \ iff \ g_j(x)$$

and $x[r]$ is the timer valuation with all $r_j := 0$.


w
S - sleep
W - wait
e - execute
a
b
s
a - arrive
b - begin execution
f - finish execution
f
e


w
s
e
a, (t=T)
t:= 0
b,(t<= D-E)
x := 0
f,(x=E)
t - measures elapsed time since proc. arrived
x - measures exec. time
T - Period
E - Execution Time
D - Realitive Deadline

Classifying the Edges

Edges are classified as either:

- Timed - Representing the progress of time.
- Controllable - An action that the scheduler can control. Ex. Starting a process, resource allocation, process preemption.
- Uncontrollable - Behavior controlled by the executing process. Such as termination.

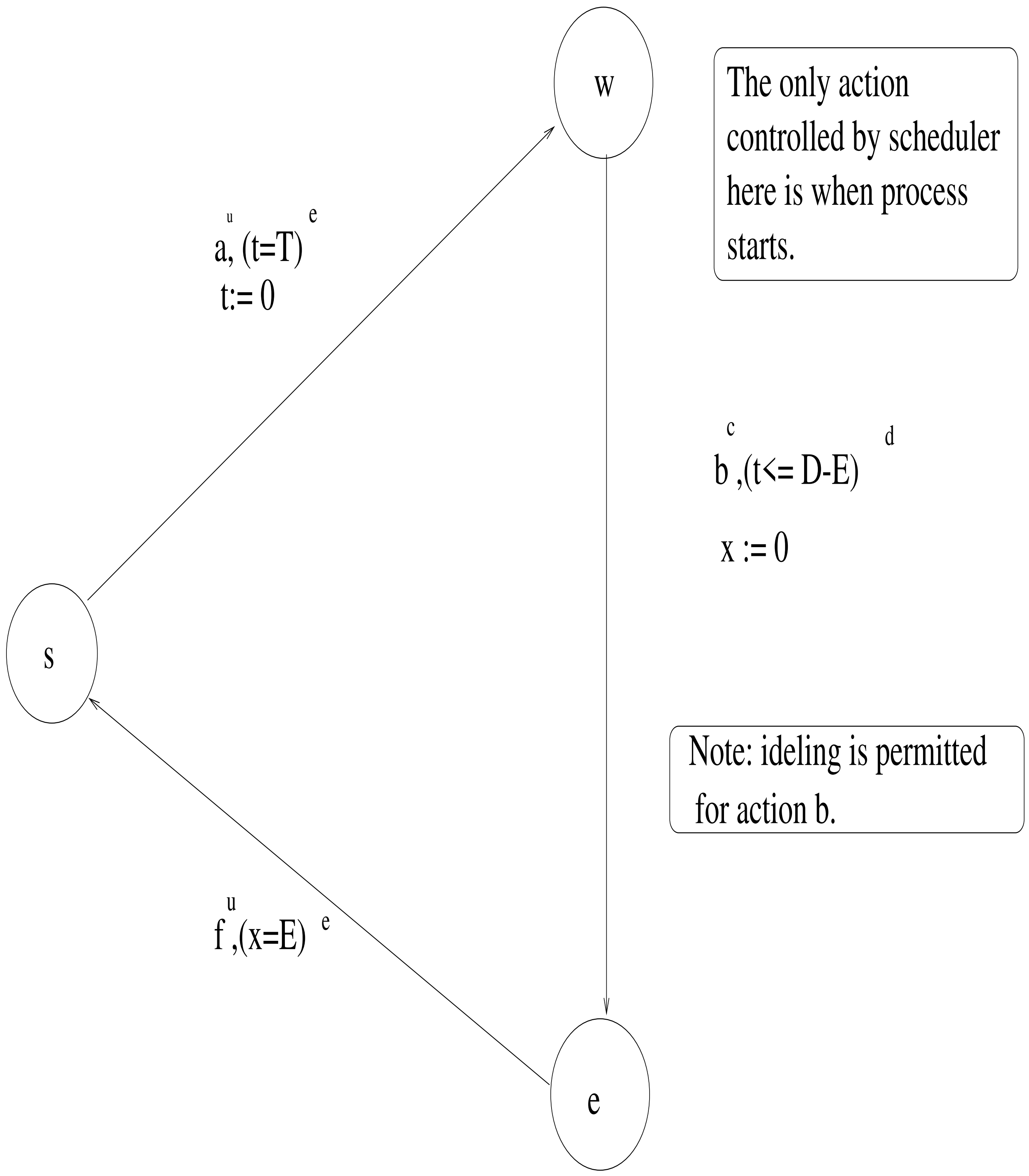
Urgency

It is important to consider when an action must be taken. In this model, time may continue to progress until an action must be taken.

- Eager Evaluation (ϵ) - Execute the action as soon as possible, ie. as soon as a guard is enabled. Time cannot progress when a guard is enabled.

- Delayable Evaluation(δ) - Can put off execution until the last moment the guard is true. So Time can't progress beyond a "falling edge" of guard.

- Lazy Evaluation (λ)- You may or may not perform the action.


w
The only action controlled by scheduler here is when process starts.
a^u, (t=T)^e
t:= 0
b^c ,(t<= D-E)^d
x := 0
s
Note: ideling is permitted for action b.
f^u ,(x=E)^e
e

Global Liveness Properties

Liveness Properties are positive in the sense that they are things we want the system to do as opposed to safety properties which are the things we don't want it to do.

For Timed Systems we are interested in:

- Timelock Freedom - In any maximum run time diverges.

- Livelock Freedom - In any run some action occurs infinitely often. Thus there is no deadlock and no infinite waiting.

- Liveness -Both Timelock and Livelock Freedom hold.

These are properties about the global state of a system. We would like to be able to build a system in a modular fashion where the property holds for each component and from that we can infer it holds for the system as a whole.

Local Properties

- Structural non-Zenoness - Any circuit of the graph has at least one clock reset and its tested against a lower bound. Thus a lower bdd to the execution of any circuit.

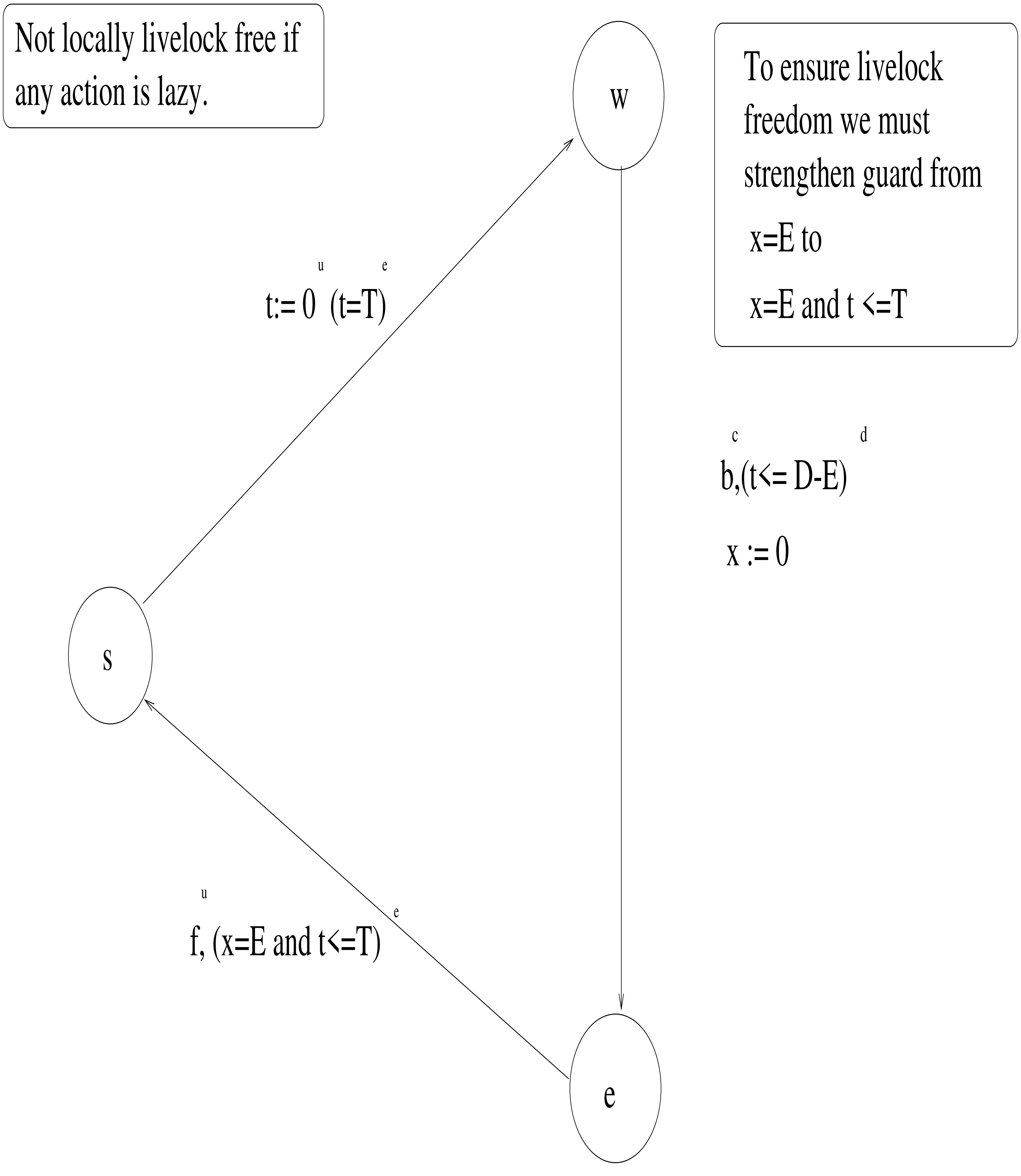
- Local Timelock Freedom - time progress stops $\Rightarrow$ a guard is enabled.

- Local Livelock Freedom - In any initial state some transition will eventually be taken.

- Structual Liveness - Structual non-Zenoness + Local Timelock Freedom + Local Livelock Freedom.

These are local properties. We can show these hold for the various components of the system.

Theorem 1 *Any locally timelock-free and locally livelock-free system is livelock-free.*


Not locally livelock free if any action is lazy.
w
To ensure livelock freedom we must strengthen guard from
x=E to
x=E and t <=T
u e
t:= 0 (t=T)
c d
b,(t<= D-E)
x := 0
s
u e
f, (x=E and t<=T)
e

Timed Priorities

When many actions are enabled we apply some priority to determine which is taken.

Restrict the guards of the lower priority actions to so as to let those actions of higher priority proceed.

A Priority order, $a_1 \prec a_2$ is a partial oder.

A Priority Rule $pr = (C^i, \prec^i)$ where C^i is a constraint and $\prec^i$ a priority order.

(TS,pr) represents a timed system with priorities, TS', obtained by restricting the guards as follows:

$$g'_j = g_j \wedge \bigwedge_{(C,\prec)\in pr} (\neg C \vee \bigwedge_{\substack{\exists i \in I \\ s_j \prec s_i}} \neg \Diamond g_i)$$

Saying that a_j is allowed iff there is no transition a_i leaving s that has priority over a_j and will become enabled.

Parallel Composition

Given two timed systems TS_1 and TS_2 their composition,$TS = TS_1||TS_2$, is defined:

$$TS = (S_1 \times S_2, A, \longrightarrow, X_1 \cup X_2, h)$$

Transitions of $\longrightarrow$ are obtained by interleaving or by synchronization ($|$).

- $h((s_1, s_2), a_1, (s_1', s_2)) = ((s_1, s_2), b_1, (s_1', s_2))$

- $h((s_1, s_2), a_1, (s_1, s_2')) = ((s_1, s_2), b_1, (s_1, s_2'))$

- $h((s_1, s_2), a_1|a_2, (s_1', s_2')) = ((s_1, s_2), b_1|b_2, (s_1', s_2'))$

The new priority of the combined system will give the synchronized action the highest priority.

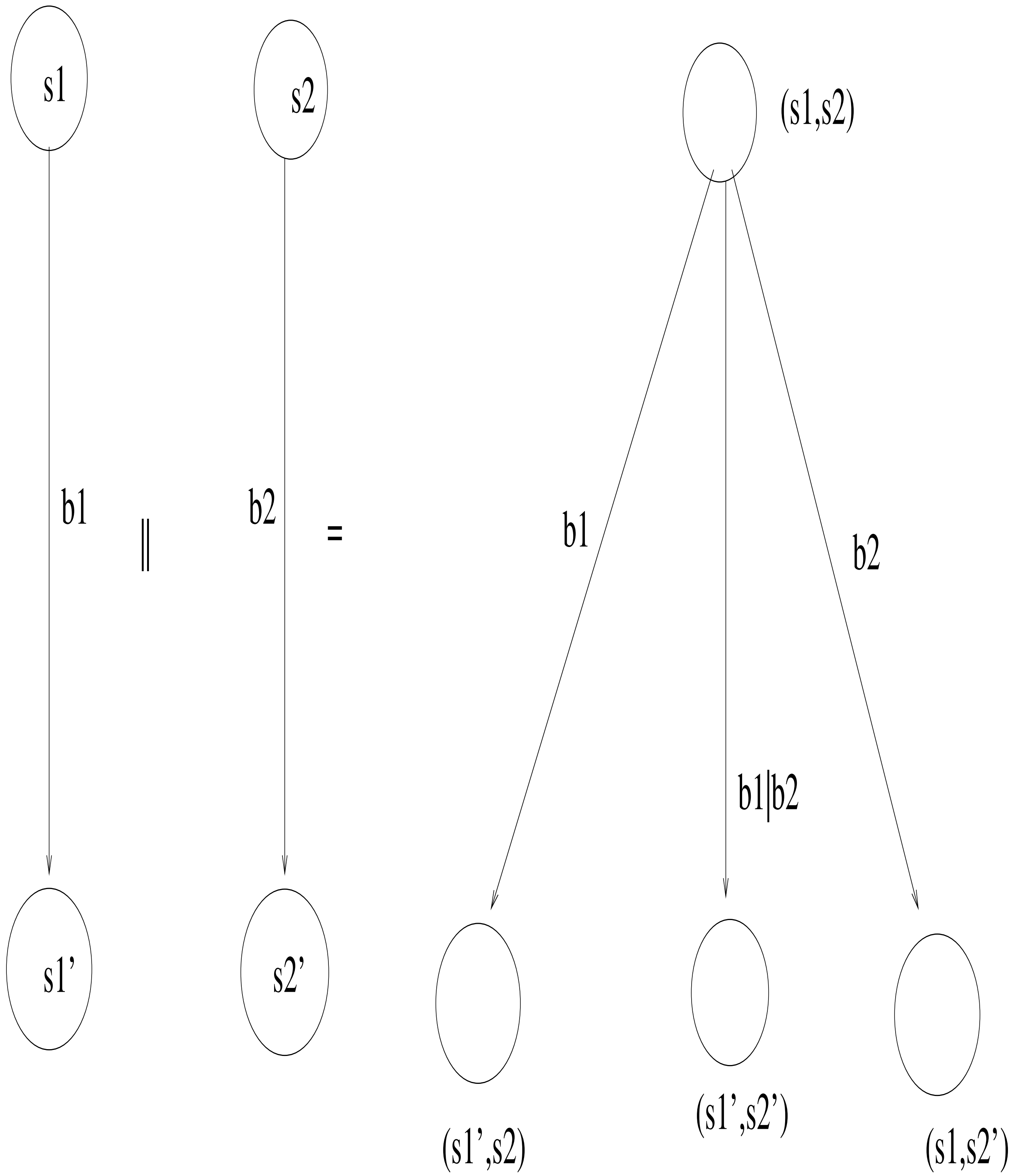
Note: if $a_1|a_2 \neq \perp$ then $b_1|b_2 = (a_1|a_2, g_1 \wedge g_2, \tau', r_1 \cup r_2$. Where we define τ' as follows:

- If at least one of the urgency constraints is eager then $\tau' = \epsilon$

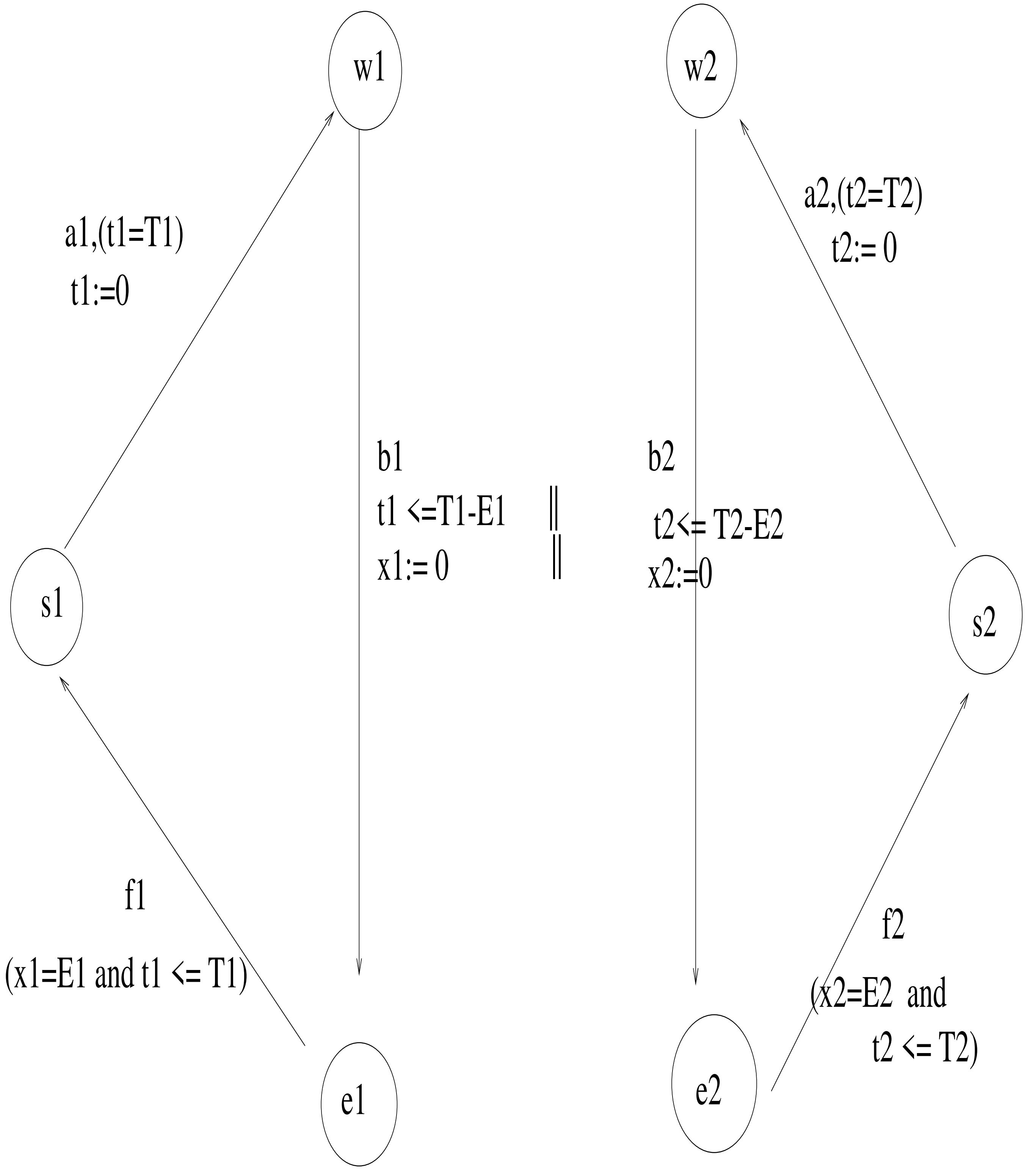
- If at least one of the urgency constraints is delayed and the other is not eager then, $\tau' = \delta$

- O/W $\tau' = \lambda$

Theorem 2 *If TS_1 and TS_2 are structurally live, then $TS_1||TS_2$ is structurally live.*



s1
s2
(s1,s2)
b1
||
b2
=
b1
b2
b1|b2
s1'
s2'
(s1',s2')
(s1',s2)
(s1,s2')


w1
w2
a1,(t1=T1)
t1:=0
a2,(t2=T2)
t2:= 0
b1
t1 <=T1-E1
x1:= 0
||
b2
t2<= T2-E2
x2:=0
s1
s2
f1
(x1=E1 and t1 <= T1)
f2
(x2=E2 and
t2 <= T2)
e1
e2

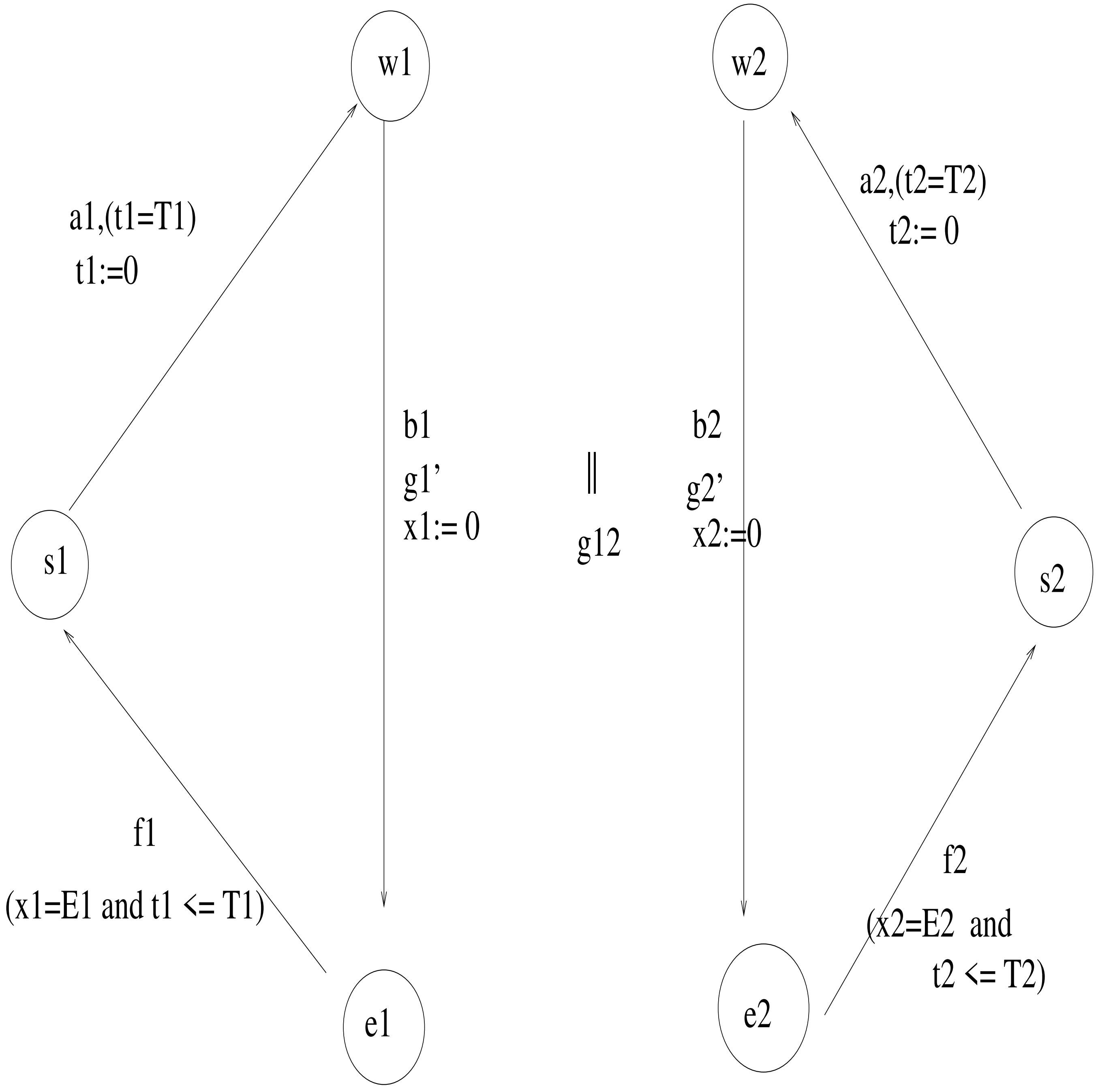
To get the processes to exchange data we wish to coordinate them via $g_1|g_2$ action whenever possible w/o violating timing constraints. The associated timed system with priorities must ensure :

$$g_{b1} \prec g_{b1}|g_{b2} \wedge g_{b2} \prec g_{b1}|g_{b2}$$

The resulting timed actions in the system with priorities will have guards:

$$\begin{aligned}
g_1' &= ((x_1 \leq T_1 - E_1) \wedge \neg\Diamond(x_2 \leq T_2 - E_2))^\delta \\
&= ((x_1 \leq T_1 - E_1) \wedge (x_2 > T_2 - E_2))^\delta \\
g_2' &= ((x_2 \leq T_2 - E_2) \wedge \neg\Diamond(x_1 \leq T_1 - E_1))^\delta \\
&= ((x_2 \leq T_2 - E_2) \wedge (x_1 > T_1 - E_1))^\delta \\
g_{12} &= ((x_1 \leq T_1 - E_1) \wedge (x_1 \leq T_2 - E_2))^\delta
\end{aligned}$$

The product system is structually live.



g1' = x1 <= T1-E1 and x2 >T2-E2 g2'=x2<=T2-E2 and x1 > T1-E1

g12 = (x1 <=T1 - E1) and (x2 <= T2 - E2)

Prelude to Scheduling

- Let K be a special constraint defined as follows:

$$\bigvee_{i=1}^{n} s_i \wedge C_i$$

where s_i is a boolean denoting presence at state s_i and C_i is our usual constraint on clocks.

- Restriction of TS by a constraint K, denoted TS/K, is the system TS with each guard of a *CONTROLLABLE* transition (s, a, g, τ, r, s') replaced by:

$$g'(x) = g(x) \wedge K(s', x[r])$$

- Proper Invariant - A constraint K is a proper invariant of TS, denoted by $TS \models \mathsf{inv}(K)$, if K is preserved by the edges of $\mathcal{E}$.

- Control Invariant - K is a control invariant if $TS/K \models \mathsf{inv}(K)$

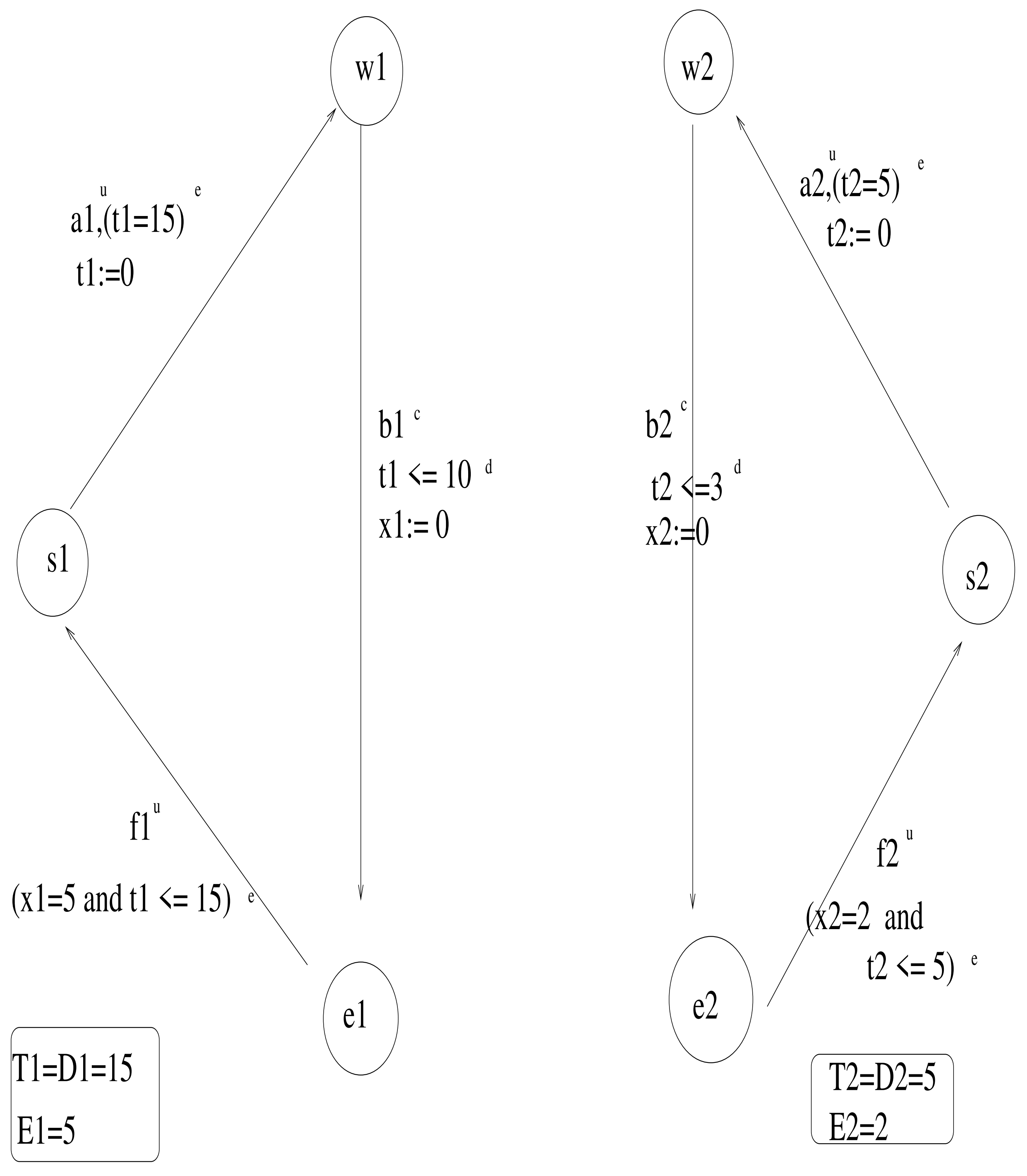
Mutual Exclusion

Consider n copies of our periodic process.

$$K_{mutex} = \bigwedge_{i \neq j} \neg e_i \vee \neg e_j$$

Since $TS^u \models \text{inv}(K_{mutex})$ we can conclude it is a control invariant.

Consider the next example:


w1
a1,(t1=15)
u
e
t1:=0
b1
c
t1 <= 10
d
x1:= 0
s1
f1
u
(x1=5 and t1 <= 15)
e
e1
T1=D1=15
E1=5
w2
a2,(t2=5)
u
e
t2:= 0
b2
c
t2 <=3
d
x2:=0
s2
f2
u
(x2=2 and
t2 <= 5)
e
e2
T2=D2=5
E2=2

We obtain TS/K_{mutex} by restricting the controllable guards g_{b1} and g_{b2} to:

$$
\begin{aligned}
g_{b1} &= (t_1 \leq D_1 - E_1) \wedge \neg e_2 = (t_1 \leq 10) \wedge \neg e_2 \\
g_{b2} &= (t_2 \leq D_2 - E_2) \wedge \neg e_1 = (t_2 \leq 2) \wedge \neg e_1
\end{aligned}
$$