Bisimilarities Induced by Relations on Actions

S. Arun-Kumar
sak@cse.iitd.ernet.in
Department of Computer Science and Engineering
I. I. T. Delhi, Hauz Khas, New Delhi 110 016.
http://www.cse.iitd.ernet.in/∼sak

September 14, 2006

Outline

Outline

Bisimulations: Vanilla flavoured

Let $\mathbf{P}$ be the set of processes defined on a set Act of actions.

Definition 1 *A binary relation $R \subseteq \mathbf{P} \times \mathbf{P}$ is a (strong) bisimulation if pRq implies the following conditions for all $a \in Act$.*

$$p \xrightarrow{a} p' \Rightarrow \exists q' : q \xrightarrow{a} q' \wedge p'Rq' \quad (1)$$

and

$$q \xrightarrow{a} q' \Rightarrow \exists p' : p \xrightarrow{a} p' \wedge p'Rq' \quad (2)$$

The largest bisimulation is bisimilarity and is an equivalence, (denoted $\sim$).

Bisimulations and Bisimilarity

- Simple and intuitively appealing theory
- Very nice algebraic properties
- Bisimilarity is the smallest equivalence relation which respects branching behaviour
- Park's induction principle
- Very efficient algorithms for proving bisimilarity of systems
- Has a nice game theoretic interpretation
- Algorithms for verification of other equivalences of concurrent systems use bisimulation

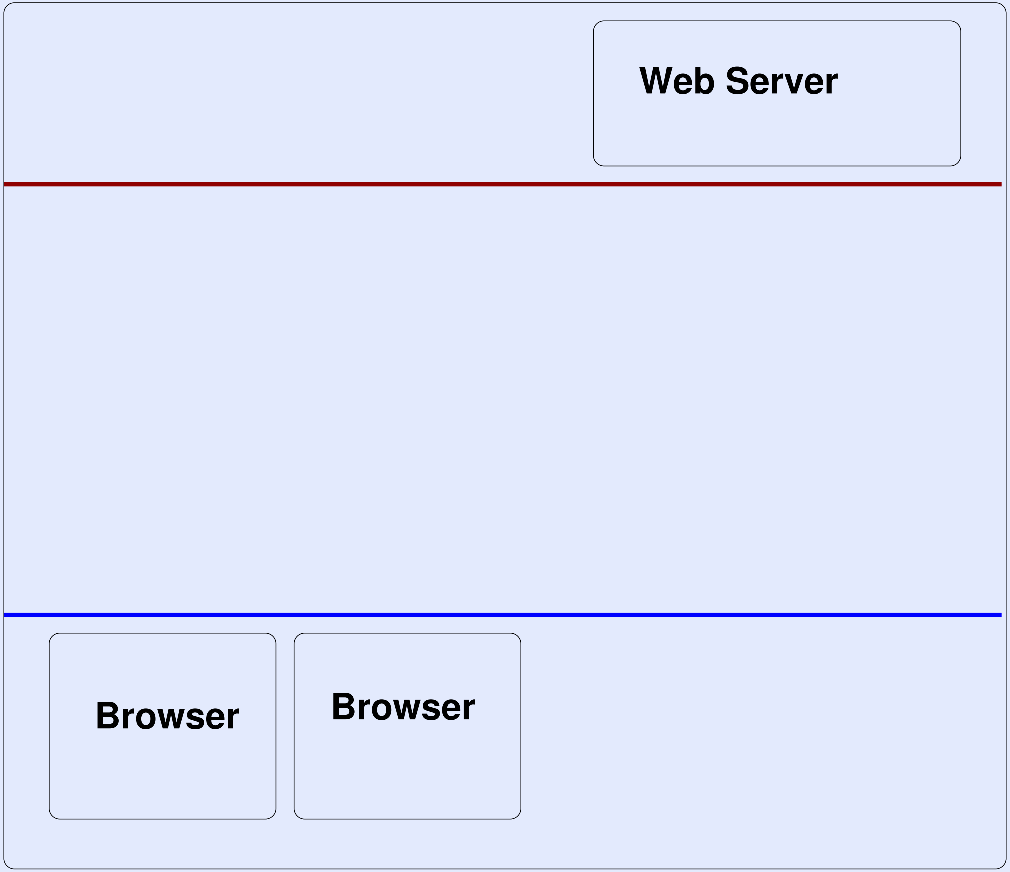
Properties of Bisimulations

- The identity relation on processes is a bisimulation
- Arbitrary unions of bisimulations are bisimulations.
- The converse of each bisimulation is also a bisimulation
- The relational composition of bisimulations is a bisimulation.
- Let $\mathscr{B}$ be a function on binary relations on $\mathbf{P}$ s.t. $\langle p, q \rangle \in \mathscr{B}(R)$ if p and q satisfy the conditions of definition 1. Then
  - $\mathscr{B}$ is monotonic i.e. $R \subseteq S \Rightarrow \mathscr{B}(R) \subseteq \mathscr{B}(S)$.
  - R is a strong bisimulation iff $R \subseteq \mathscr{B}(R)$.
  - If R is a strong bisimulation then so is $\mathscr{B}(R)$.
  - $\sim = \bigcup\{R | R \subseteq \mathscr{B}(R)\}$ is the largest fixpoint of $\mathscr{B}$.
- $\sim$ is the largest bisimulation and an equivalence relation on processes

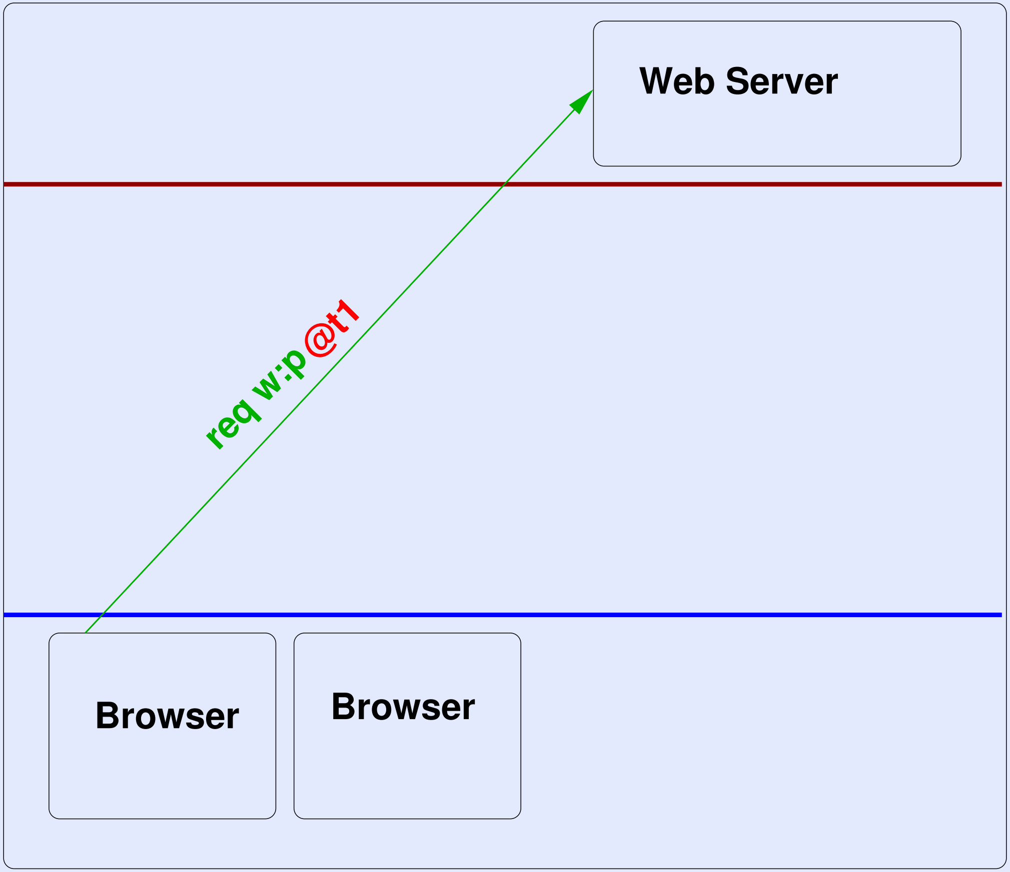
Outline

- Vanilla Bisimulations
- Example
- The basic framework
- Bisimulations: Generalisation
- Inheritance
- The Proxy Revisited
- An On-the-fly Algorithm
- Conclusions

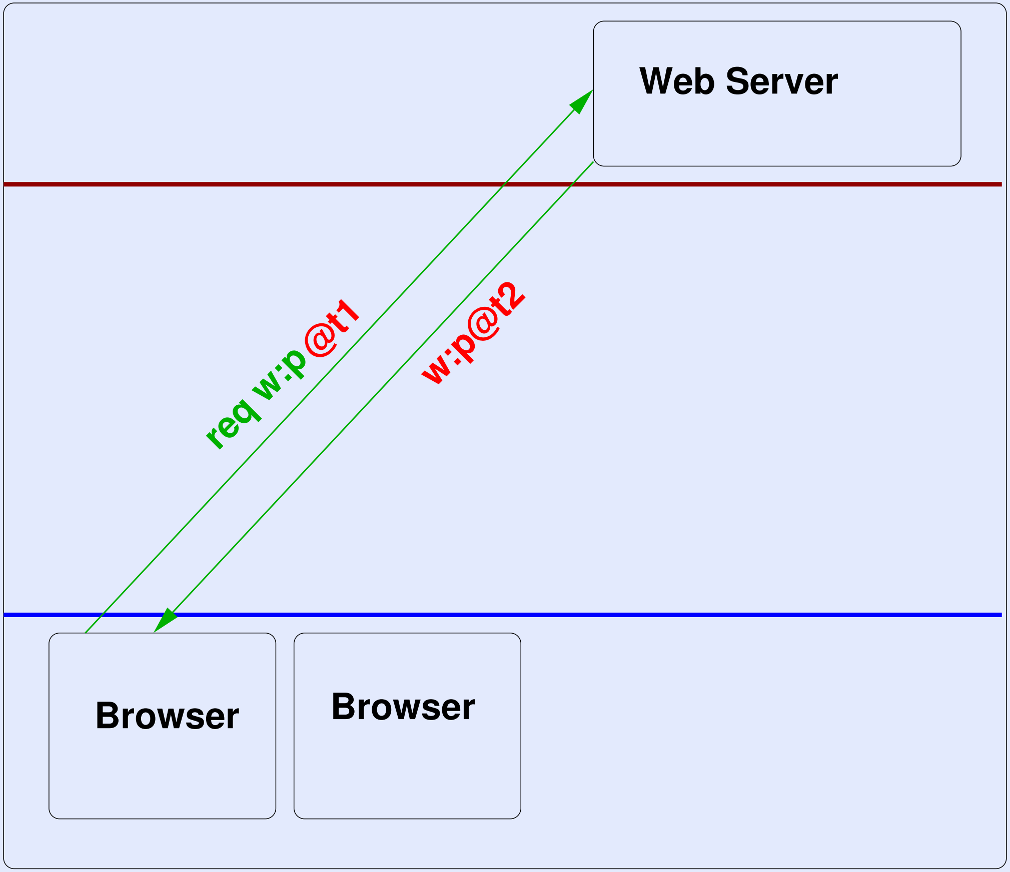
Example: Browser



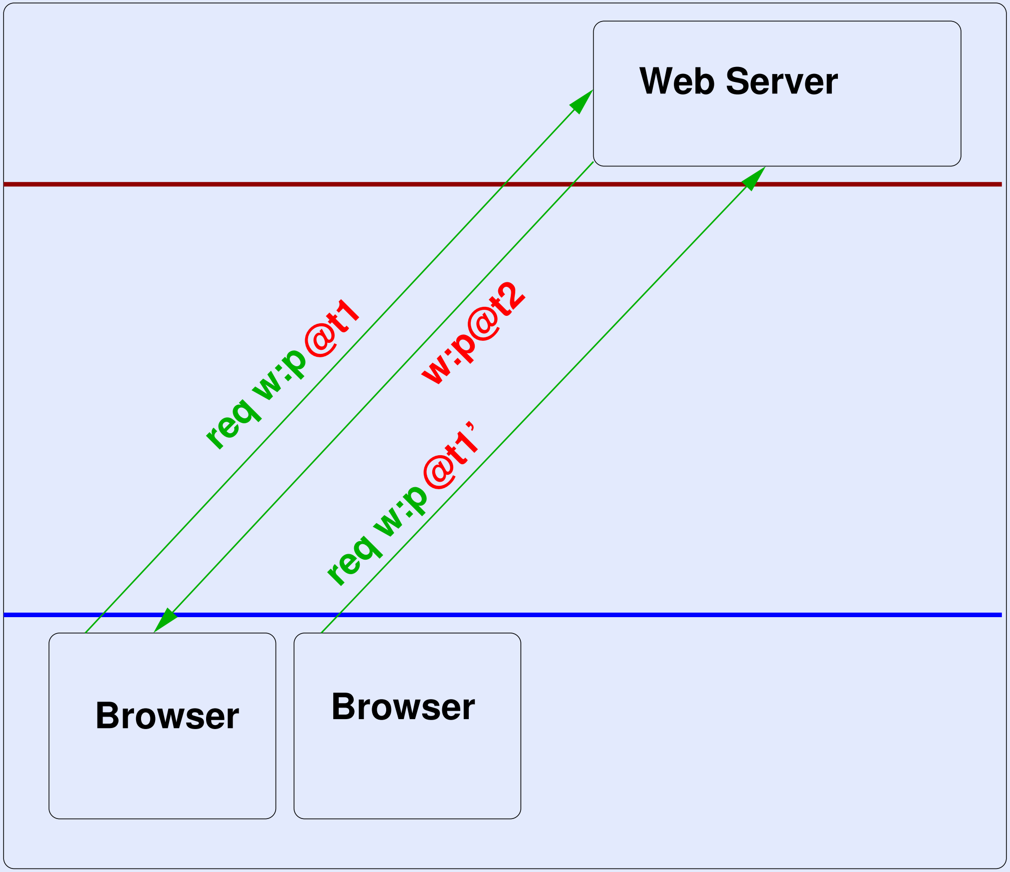
Example: Browser



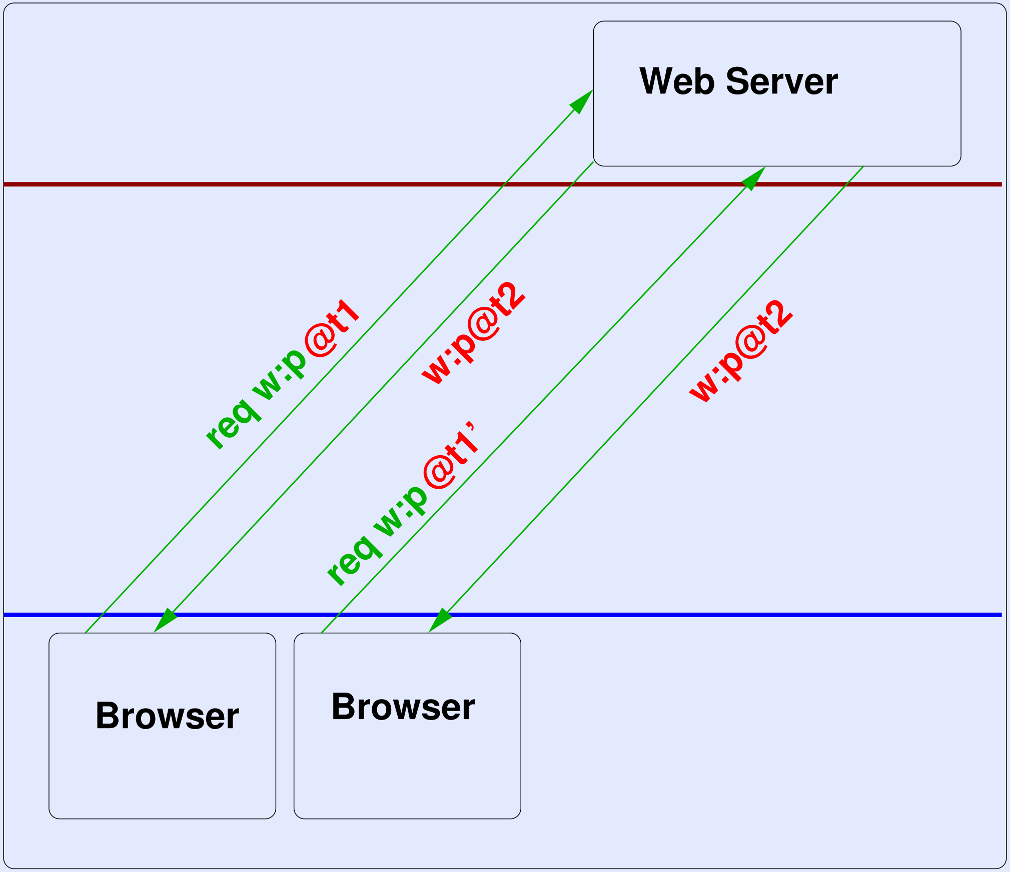
Example: Browser



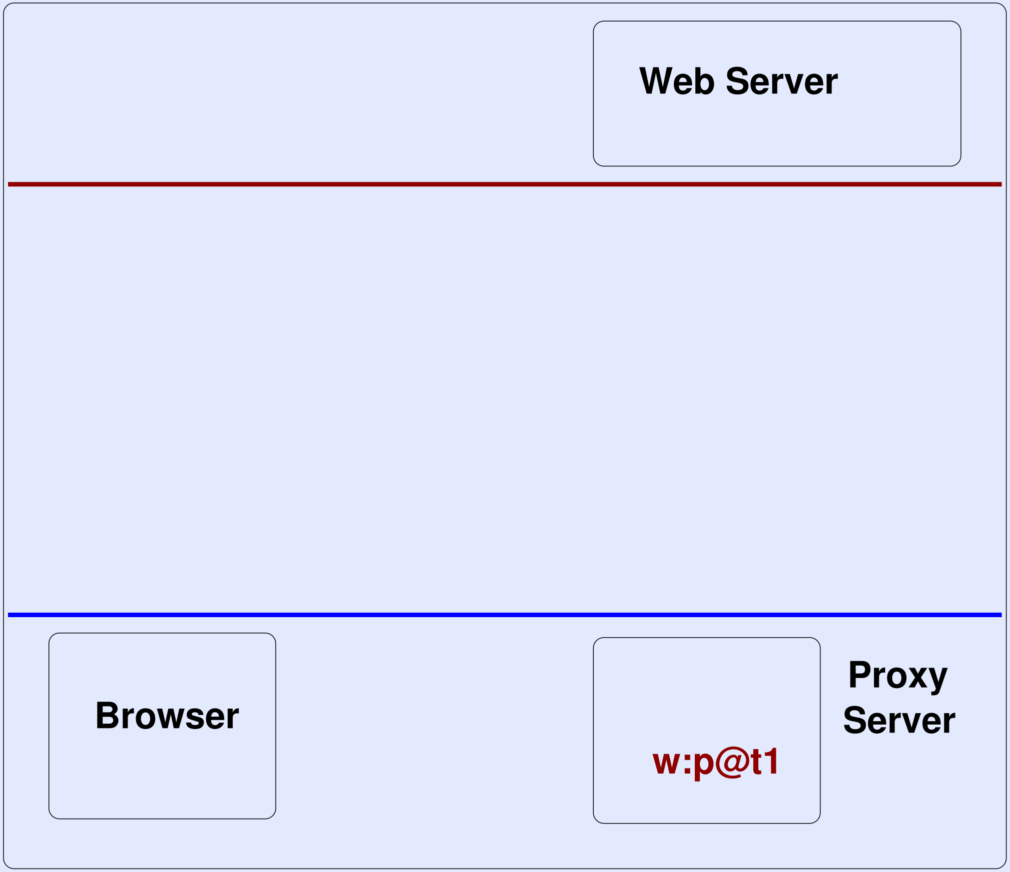
Example: Browser



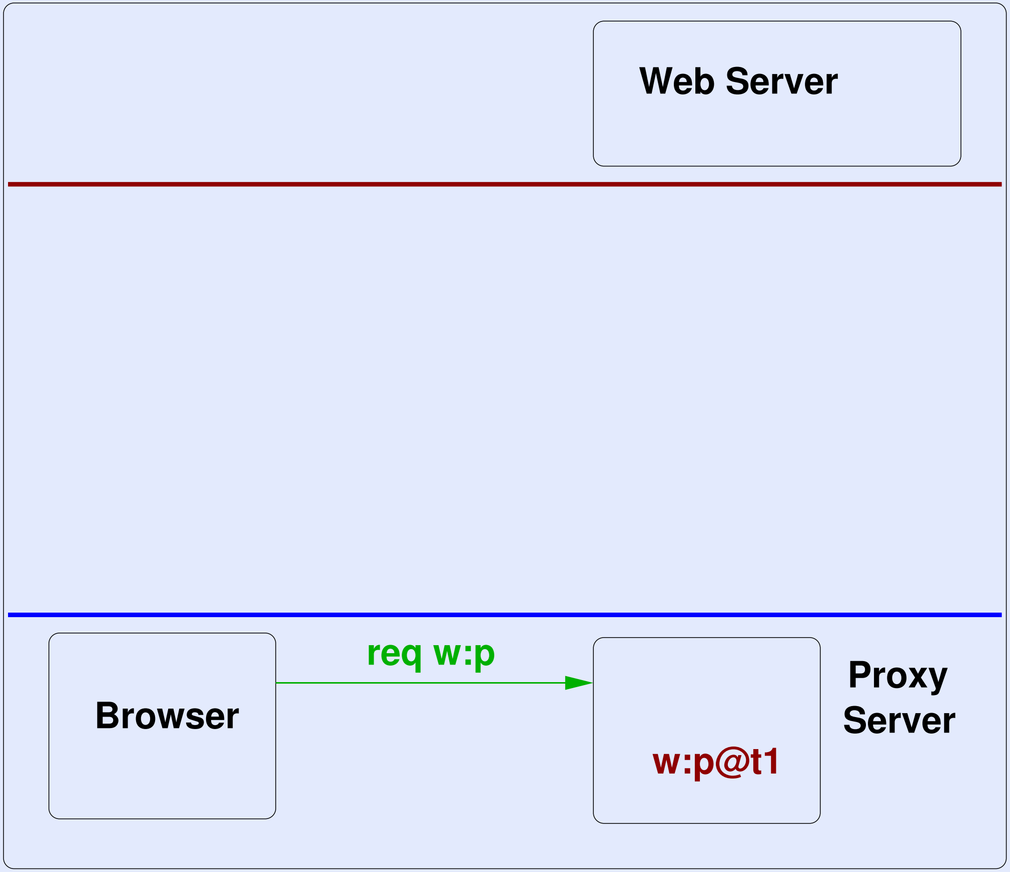
Example: Browser



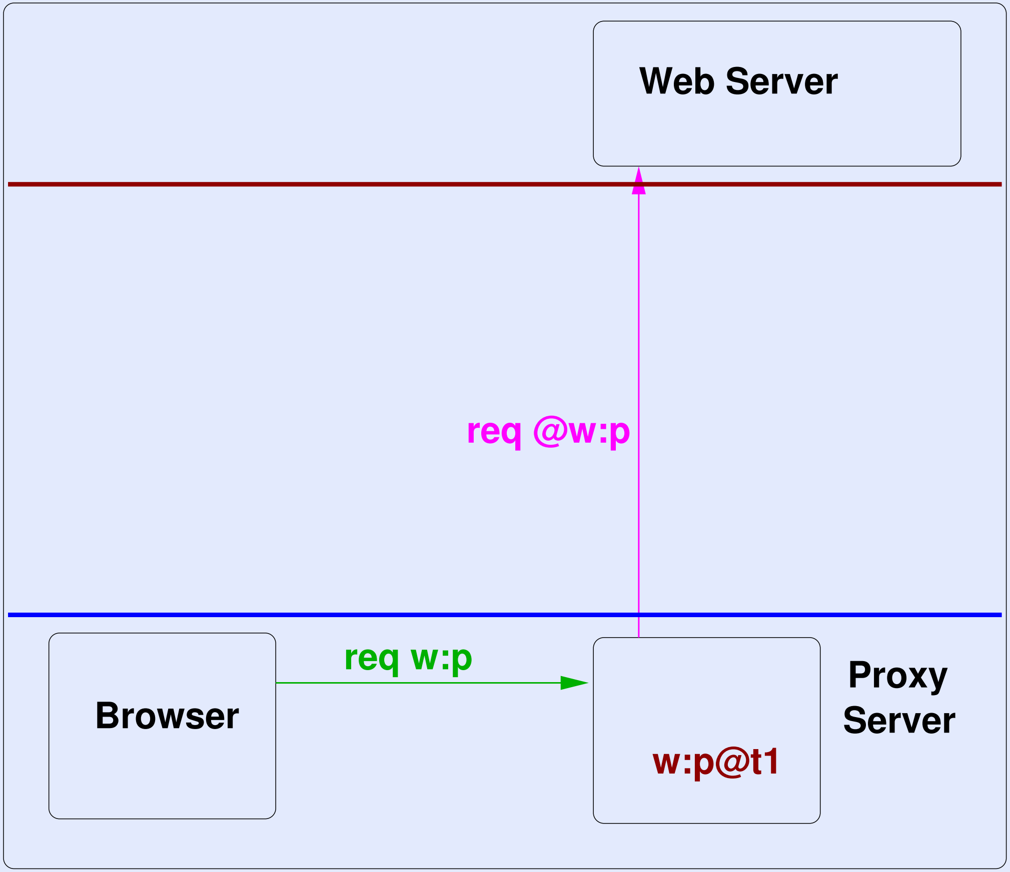
Example: Proxy Server



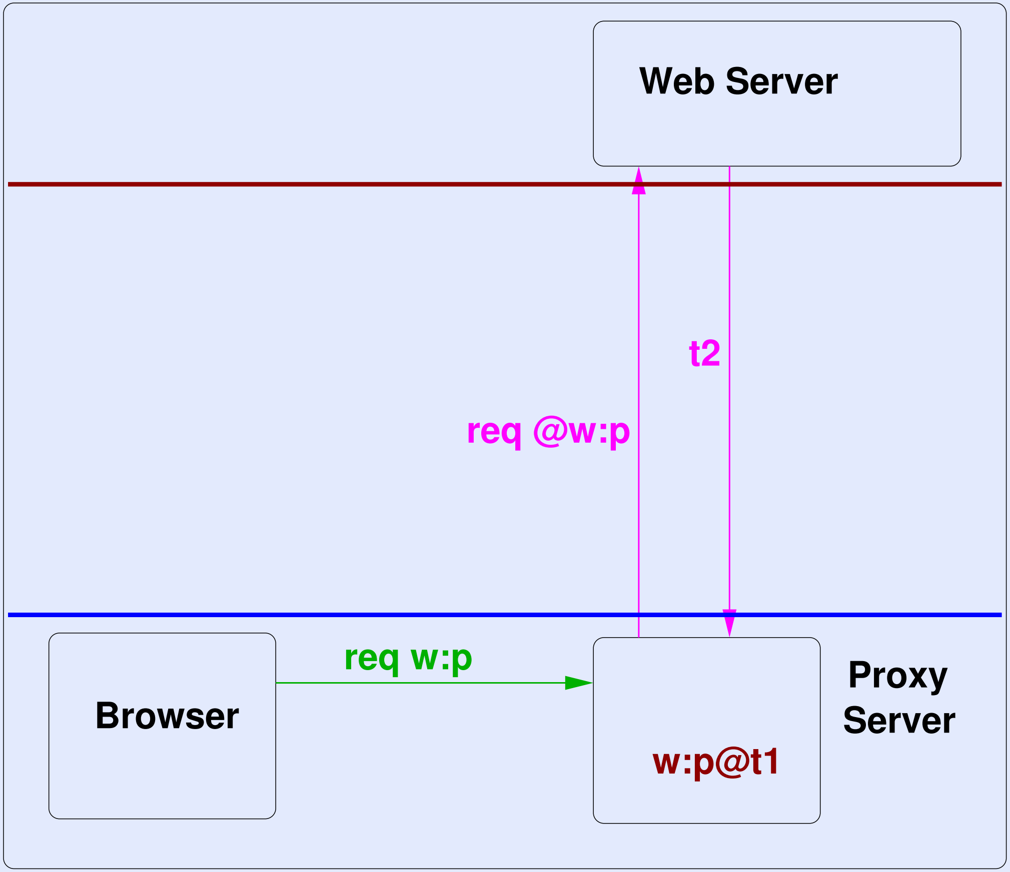
Example: Proxy Server



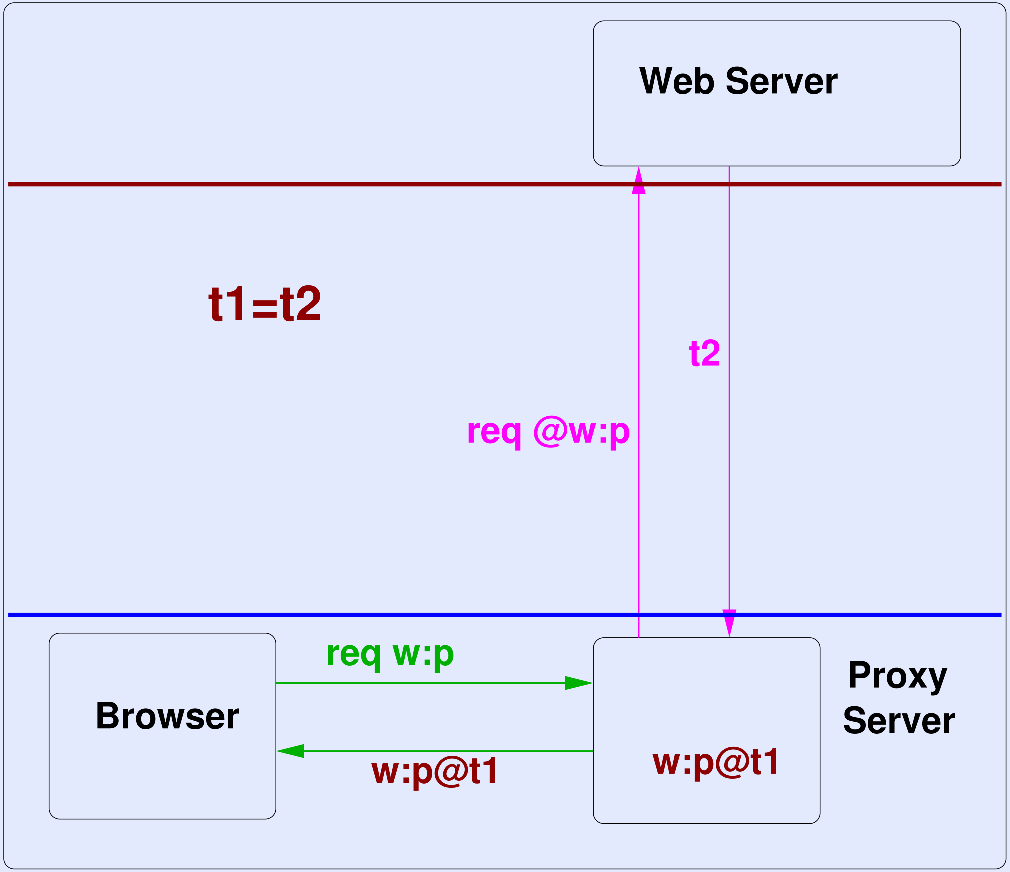
Example: Proxy Server



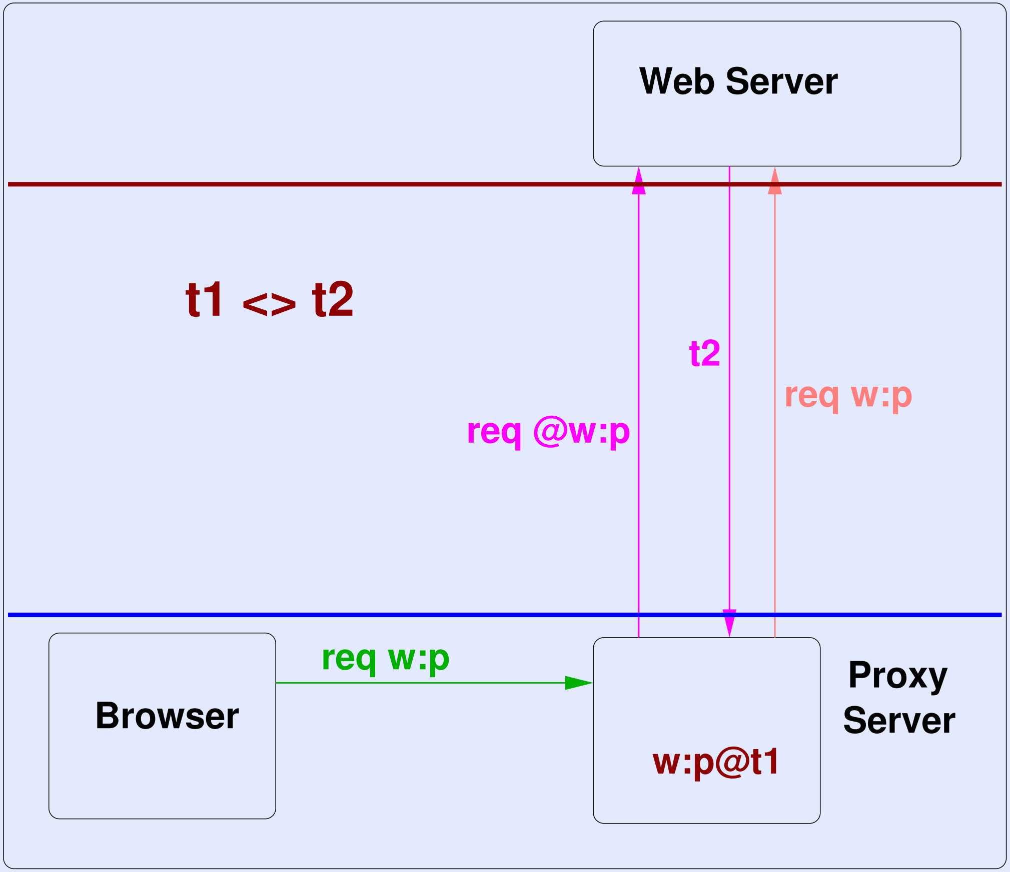
Example: Proxy Server



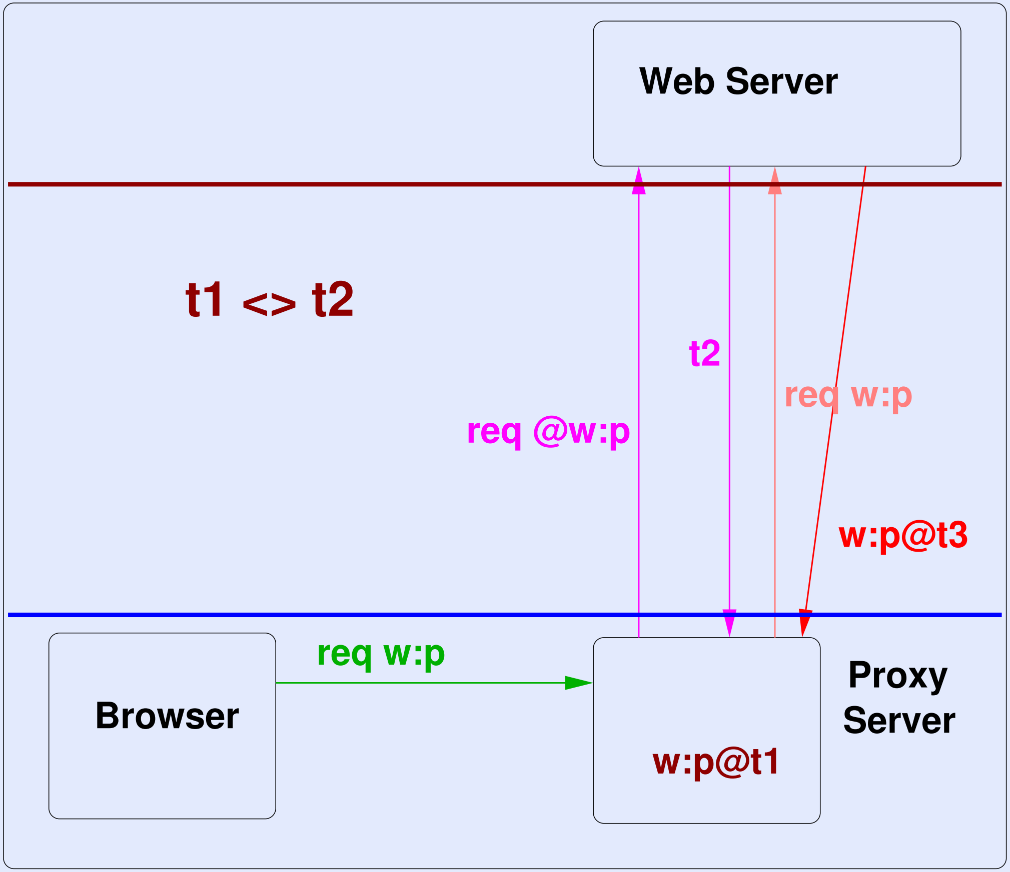
Example: Proxy Server



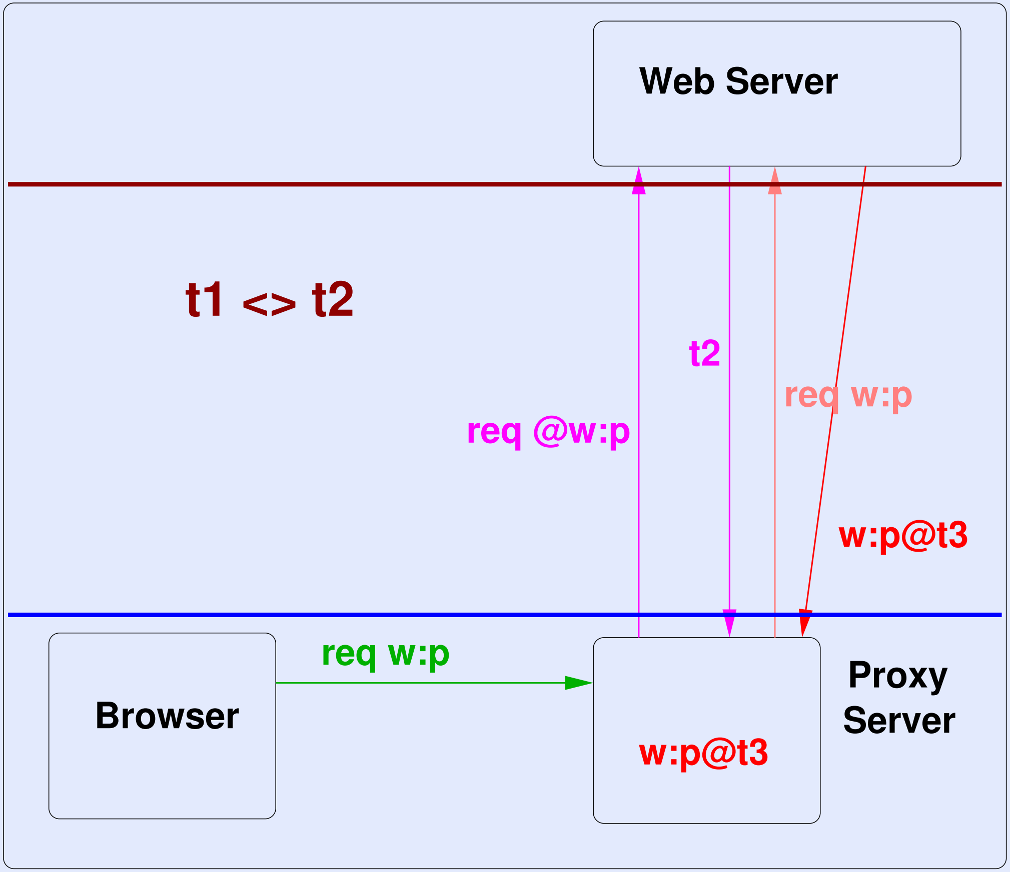
Example: Proxy Server



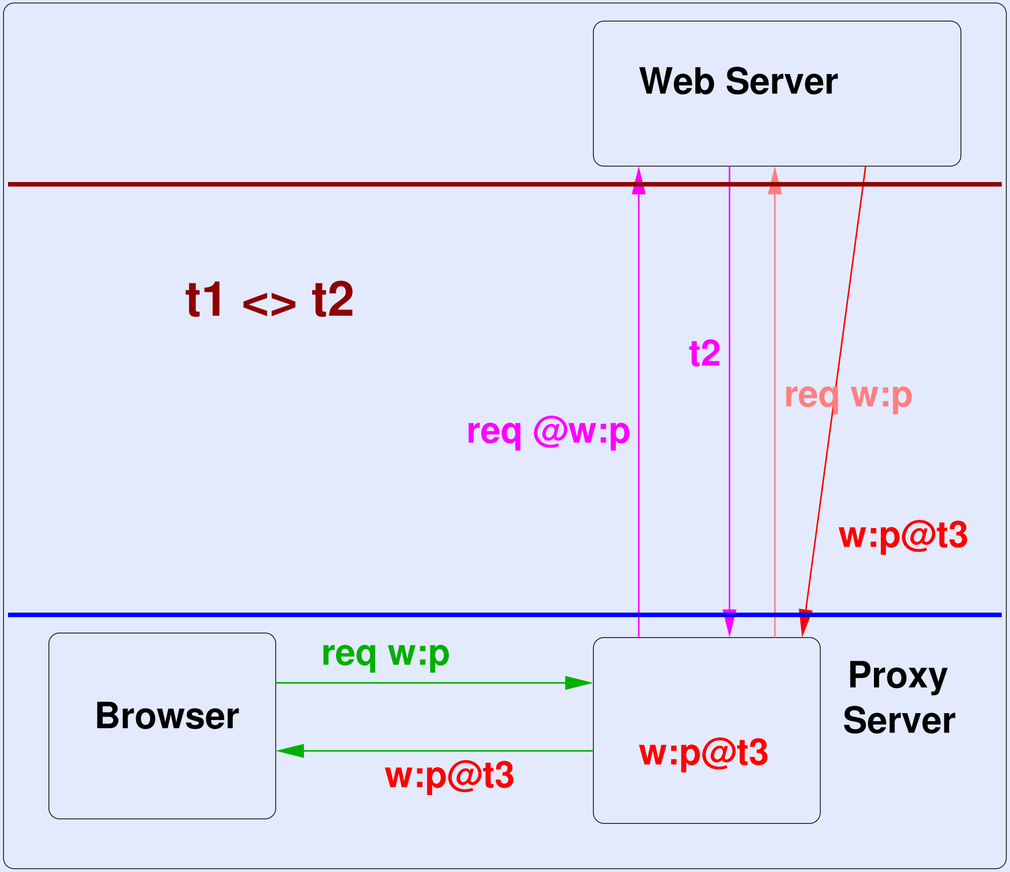
Example: Proxy Server



Example: Proxy Server



Example: Proxy Server



Modelling in CCS: Actions

The action set.

$gp()$	–	*g*et *p*age
$op(a)$	–	*o*utput *p*age a on screen
$drp()$	–	*d*irect *r*equest for *p*age
$dsp(h, a)$	–	*d*irectly *s*erve *p*age
$irp()$	–	*i*ndirect *r*equest for *p*age
$isp(h, a)$	–	*i*ndirectly *s*erve *p*age
$drh()$	–	*d*irect *r*equest for *h*eader
$dsh(h)$	–	*d*irectly *s*erve *h*eader

Modelling in CCS

A typical client $\textsc{DClient}$, which accesses the web-server *directly*, has the following definition.

$$\textsc{DClient} \triangleq gp().\overline{drp()}.dsp(h,a).\overline{op(a)}.\textsc{DClient}$$

With the introduction of a proxy server, the clients communicate only with the proxy.
The actions involving communications of the clients with proxy server are irp and isp.

$$\textsc{IClient} \triangleq gp().\overline{irp()}.isp(h,a).\overline{op(a)}.\textsc{IClient}$$

Modelling in CCS: Proxy

- Assume it serves only one request at a time
- Initial undefined content $(\bot, \bot)$ in cache
- On the first request it obtains the full page from the web-server.
- For each subsequent request it merely sends a request with the header h_0 as parameter and waits in the state $\textsc{PrWait}(h_0, a_0)$, where (h_0, a_0) is the current content in its cache.

$$
\begin{array}{lcl}
\textsc{Proxy0}(\bot, \bot) & \triangleq & irp().\textsc{ReqPage}(\bot, \bot) \\
\textsc{Proxy}(h_0, a_0) & \triangleq & irp().\textsc{ClWait}(h_0, a_0) \\
\textsc{ReqPage}(h_0, a_0) & \triangleq & \overline{drp()}.\textsc{ReqSent}(h_0, a_0) \\
\textsc{ClWait}(h_0, a_0) & \triangleq & \overline{drh(h_0)}.\textsc{PrWait}(h_0, a_0)
\end{array}
$$

Modelling in CCS: Proxy

- Web-server may respond by sending back the same header h_0 (indicating no change in page content), or
- Send an updated page content (h'_0, a'_0), with $h'_0 \neq h_0$. The proxy caches this new content.
- Its cache has the latest content on demand.

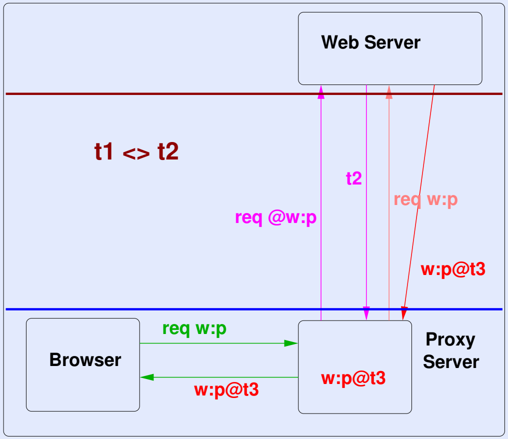
$$
\begin{array}{lcl}
\textsc{PrWait}(h_0, a_0) & \triangleq & dsh(h_0).\textsc{Cached}(h_0, a_0) + \\
 & & dsp(h'_0, a'_0).\textsc{Cached}(h'_0, a'_0) \\
\textsc{ReqSent}(h_0, a_0) & \triangleq & dsp(h''_0, a''_0).\textsc{Cached}(h''_0, a''_0) \\
\textsc{Cached}(h, a) & \triangleq & \overline{isp(h, a)}.\textsc{Proxy}(h, a)
\end{array}
$$

The client-proxy system in the local area network is defined as follows:

$$
\begin{array}{lcl}
\textsc{CPSys} & \triangleq & (\textsc{IClient} | \textsc{Proxy0}(\bot, \bot)) \\
 & & \backslash \{irp(\_, \_), isp(\_, \_)\}
\end{array}
$$

_ denote wildcard values.

Example: Proxy Server: Analysis



- CPSys and DClient not weakly bisimilar, since CPSys may perform actions such as $dsh(\_) \notin Sort(\text{DClient})$.
- However they are both *functionally equivalent* in an obvious sense.

Outline

- Vanilla Bisimulations
- Example
- The basic framework
- Bisimulations: Generalisation
- Inheritance
- The Proxy Revisited
- An On-the-fly Algorithm
- Conclusions

Labelled Transition Systems

Definition 2 *A labelled transition system (LTS) is a 3-tuple* $\langle \mathbf{P}, Act, \longrightarrow, \mathbf{I} \rangle$ *where*

- $\mathbf{P}$ *is a set of process states*
- Act *is a set of actions and* $\longrightarrow \subseteq \mathbf{P} \times Act \times \mathbf{P}$ *is the transition relation.*
- $\mathbf{I} \subseteq \mathbf{P}$ *is a set of initial states*

A LTS $\langle \mathbf{P}, Act, \longrightarrow, \mathbf{I} \rangle$ may equally well be viewed as a structure $\langle \mathbf{P}, Act^*, \longrightarrow, \mathbf{I} \rangle$. The usual theory of bisimulations does not distinguish between the two.

LTSs and Processes

Definition 3 *A rooted LTS* $\langle \mathbf{P}, Act, \longrightarrow, p_0\rangle$ *is a LTS* $\langle \mathbf{P}, Act, \longrightarrow, \{p_0\}\rangle$ *with a single initial state* p_0.

- $p \xrightarrow{a} q$ denotes $(p, a, q) \in \longrightarrow$
- a-Successors of p: $p \xrightarrow{a} = \{q \mid p \xrightarrow{a} q\}$
- Successors of p: $p \longrightarrow = \bigcup_{a \in Act} p \xrightarrow{a}$
- Derivatives of p: $p \longrightarrow^* = \{p\} \cup \bigcup_{q \in p \longrightarrow} q \longrightarrow^*$
- q is reachable from p: $q \in p \longrightarrow^*$
- Process p: (sub-)LTS rooted at state p and consisting of all the states and transitions reachable from p.

Outline

- Example
- The basic framework
- Bisimulations: Generalisation
- Inheritance
- The Proxy Revisited
- An On-the-fly Algorithm
- Conclusions

(ρ, σ)-Bisimulations

- **Definition 4**
  - $\mathbf{P}$: *the set of states*
  - *ρ and σ: binary relations on Act*

    .
  - $R \subseteq \mathbf{P} \times \mathbf{P}$

  R is a (ρ, σ)-induced bisimulation or simply a (ρ, σ)-bisimulation if pRq implies the following conditions.

  $$\forall a \in Act[p \xrightarrow{a} p' \Rightarrow \exists b, q' : a\rho b \wedge q \xrightarrow{b} q' \wedge p'Rq'] \quad (3)$$
  $$\forall b \in Act[q \xrightarrow{b} q' \Rightarrow \exists a, p' : a\sigma b \wedge p \xrightarrow{a} p' \wedge p'Rq'] \quad (4)$$

- The largest (ρ, σ)-bisimulation (under set containment) is called (ρ, σ)-bisimilarity and denoted $\sqsubseteq_{(\rho,\sigma)}$.
- A $(=, =)$-bisimulation will be called a natural bisimulation.

(ρ, σ)-Bisimulations: Smooth Generalization

- Arbitrary unions of (ρ, σ)-bisimulations are (ρ, σ)-bisimulations.
- Let $\mathscr{B}_{(\rho,\sigma)}$ be a function on binary relations on $\mathbf{P}$ s.t. $\langle p, q\rangle \in \mathscr{B}_{(\rho,\sigma)}(R)$ iff p and q satisfy the conditions of 4. Then:
  - $\mathscr{B}_{(\rho,\sigma)}$ is monotonic i.e.

$$R \subseteq S \Rightarrow \mathscr{B}_{(\rho,\sigma)}(R) \subseteq \mathscr{B}_{(\rho,\sigma)}(S)$$

  - R is a (ρ, σ)-bisimulation iff $R \subseteq \mathscr{B}_{(\rho,\sigma)}(R)$.
  - If R is (ρ, σ)-bisimulation then so is $\mathscr{B}_{(\rho,\sigma)}(R)$.
  - $\underline{\Box}_{(\rho,\sigma)} = \bigcup\{R \mid R \subseteq \mathscr{B}_{(\rho,\sigma)}(R)\}$ is the largest fixpoint of $\mathscr{B}_{(\rho,\sigma)}$.

Park's Induction Principle

Theorem 0.1 **(Park's Induction Principle).** *Let R be a binary relation on processes satisfying the following conditions for all pRq and $a, b \in Act$:*

$$\forall p'[p \xrightarrow{a} p' \Rightarrow \exists b, q'[a\rho b \wedge q \xrightarrow{b} q' \wedge p'(R \cup \sqsubseteq_{(\rho,\sigma)})q']]$$

$$\forall q'[q \xrightarrow{b} q' \Rightarrow \exists a, p'[a\sigma b \wedge p \xrightarrow{a} p' \wedge p'(R \cup \sqsubseteq_{(\rho,\sigma)})q']]$$

Then $R \subseteq \sqsubseteq_{(\rho,\sigma)}$.

- To prove $p \sqsubseteq_{(\rho,\sigma)} q$ it suffices to find a (ρ, σ)-bisimulation containing $\langle p, q \rangle$.
- $R : p \sqsubseteq_{(\rho,\sigma)} q$ to denote that R is a (ρ, σ)-bisimulation containing the pair $\langle p, q \rangle$.

Monotonicity

Extending $\subseteq$ and $\subset$ pointwise to pairs of relations

- $(\rho, \sigma) \subseteq (\rho', \sigma') \Rightarrow$ every (ρ, σ)-bisimulation is also a (ρ', σ')-bisimulation
- And

$$(\rho, \sigma) \subseteq (\rho', \sigma') \Rightarrow \sqsubseteq_{(\rho,\sigma)} \subseteq \sqsubseteq_{(\rho',\sigma')}$$

- But

$$(\rho, \sigma) \subset (\rho', \sigma') \not\Rightarrow \sqsubseteq_{(\rho,\sigma)} \subset \sqsubseteq_{(\rho',\sigma')}$$

Reflexivity and Transitivity

- If both ρ and σ are reflexive then so is $\sqsubseteq_{(\rho,\sigma)}$.
- The identity relation $\mathcal{I}$ is a (ρ, σ)-bisimulation iff both ρ and σ are reflexive
- If both ρ and σ are transitive then so is $\sqsubseteq_{(\rho,\sigma)}$
- For any (ρ, σ)-bisimulations R and S,
  $R \circ S$ is a (ρ, σ)-bisimulation iff ρ and σ are both transitive

Outline

Preorders and Partial Orders

- If both ρ and σ are preorders then $\sqsubseteq_{(\rho,\sigma)}$ is a preorder.
- If both ρ and σ are partial orders then $\sqsubseteq_{(\rho,\sigma)}$ may not be a partial order (but is guaranteed to be a preorder)
- $\sqsubseteq_{(\rho,\sigma)}$ inherits/preserves the preordering properties of ρ and σ.

Symmetry

- If both ρ and σ are symmetric then the converse of a (ρ, σ)-bisimulation is a (σ, ρ)-bisimulation.
- $\underline{\Box}_{(\rho,\rho)}$ is symmetric if ρ is symmetric
- For any ρ, if
  R is a (ρ, ρ)-bisimulation implies R^{-1} is also (ρ, ρ)-bisimulation
  then ρ must be symmetric.
- $\underline{\Box}_{(\rho,\rho)}$ is an equivalence iff ρ is an equivalence.

When $\rho = \sigma$ or $\sigma = \rho^{-1}$

Theorem 0.2 . *For any binary relation ρ on Act,*

1. $\underline{\square}_{(\rho,\rho)}$ *is a preorder iff ρ is a preorder.*
2. $\underline{\square}_{(\rho,\rho)}$ *is an equivalence iff ρ is an equivalence relation.*
3. *If ρ is a preorder then $\underline{\square}_{(\rho,\rho^{-1})}$ is an equivalence.*
4. *A strong bisimulation is simply a $(=,=)$-bisimulation.*
5. *A weak bisimulation is simply a $(\widehat{=},\widehat{=})$-bisimulation, where $\widehat{=}$ is the equivalence on strings of Act^* defined by ignoring all occurrences of the silent action τ.*

Outline

The Proxy Revisited

Let ρ relate "similar" actions. Let $=_\rho$ be the smallest equivalence such that

- $\overline{drh()} \;=_\rho\; \overline{drp()}$ and
- $dsh(h) =_\rho dsp(h,a)$, for any (h,a)
- $\varepsilon \;=_\rho \tau$

We can show that $\mathrm{CPS{\small YS}} \sqsubseteq_{(=_\rho,=_\rho)} \mathrm{DC{\small LIENT}}$.

Proxy server
Actions

The Proxy Revisited Again

Using the relative costs of different actions.

- The internal actions and getting header information cost almost nothing.
- The costliest action is receiving the entire page from the web-server.

Let $\leq$ be the smallest preorder on actions, satisfying

- $\overline{drh(h)} \leq \overline{drp()}$ and $\overline{drp()} \leq \overline{drh(h)}$, for any header h
- $dsh(h) \leq dsp(h, a)$, for any (h, a)
- $\varepsilon \leq \tau$, and $\tau \leq \varepsilon$.

Then $\text{CPS}\textsc{ys} \sqsubseteq_{(\leq,\leq)} \text{DC}\textsc{lient}$

Proxy server
Actions

Home Page
Title Page
Page 42 of 47
Go Back
Full Screen
Close
Quit

Outline

An On-the-fly Algorithm

- An adaptation of Fernandez and Mounier's on-the-fly algorithm for vanilla bisimulation
- Uses the technique of a Partial depth-first search and runs in $O(n^2|Act|)$ time to reduce backtracking.
- The adaptation uses bit arrays to store information obtained at each point about their relationship, assuming initially that they are related unless proven otherwise in the future. But requires twice the amount of information to be stored for ρ and σ.
- Assuming that both ρ and σ are available as table lookups and take no time to compute, our algorithm runs in $O(n^2|Act|^2)$ time and has a space requirement of $O(n + |Act|^2)$.

Outline

Conclusions

- A smooth theory is obtained by parametrizing bisimulations on actions.
- Bisimulations inherit their relational algebraic properties from properties of the underlying relation on actions.
- Name equality does not necessarily capture "functional similarity".
- There is a need to look at more generalized notions of equivalence based on functional similarities in behaviour.
- For open systems and for being able to prove properties locally, more general notions may be required.
- While adaptation of the on-the-fly algorithm was easy, the same cannot be said of the partitioning algorithm of Paige and Tarjan, which requires an *equivalence* relation on actions.