Behavioural Contracts for Components

Cyril Carrez

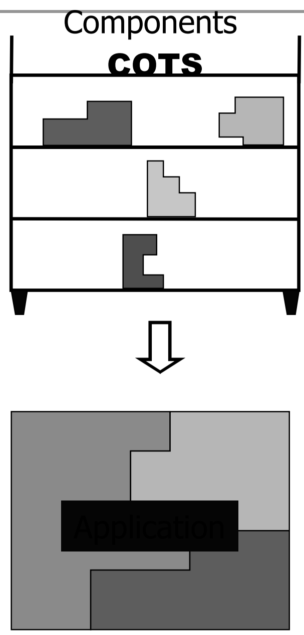
01/03/2004

NTNU

Design by Assembly

- ADL (90's)
  - components
  - connectors
  - configuration
- UML 2.0 (2003)

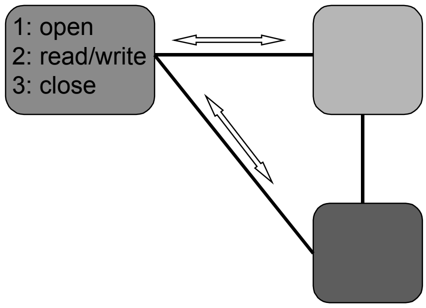
Classification
[Medvidovic & Taylor]



- Behavioural typing with explicit types
  - Regular types [Nierstrasz]
  - «non understood message» [Najm et al.]
- Contracts
  - Design by Contract [Meyer]
  - Classification [Beugnard et al.]
    - Syntactic / **behaviour** (pre/post) / **synchronisation** / QoS

Framework of the study

- Components
  - specification + code
- *Non uniform services*
- Dynamic links



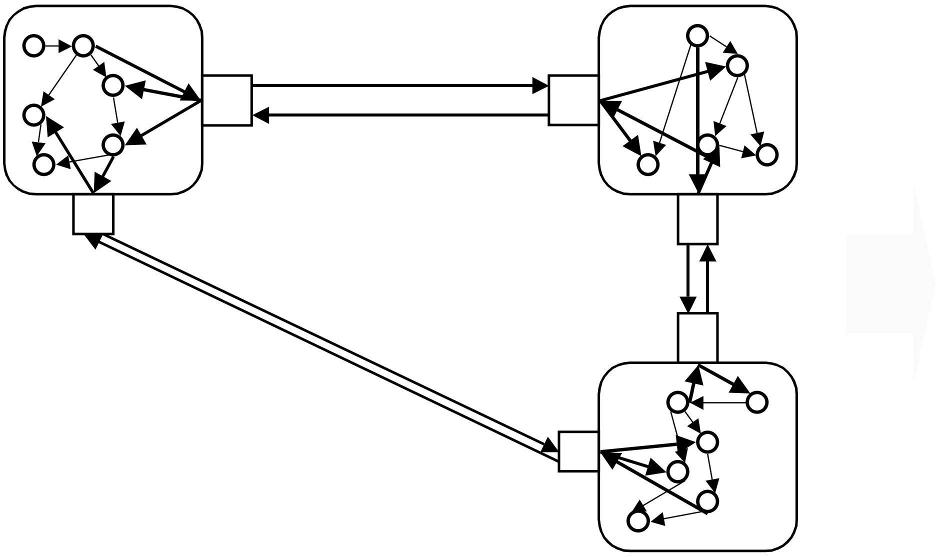
Objectives

- Safety properties: no external deadlock
- Liveness properties: messages will be consumed

Roadmap

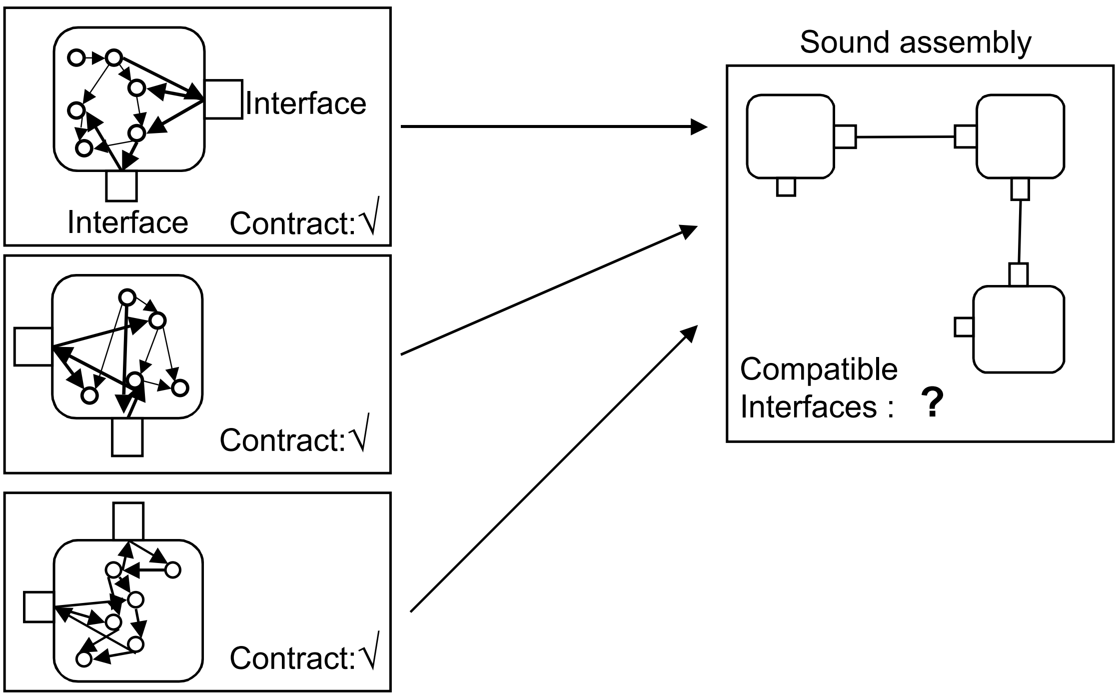
- The approach
- Interface language
- Component semantics
- Contract respect
- Sound assembly
- Conclusion & Perspectives

Approaches: Darwin, Wright,...

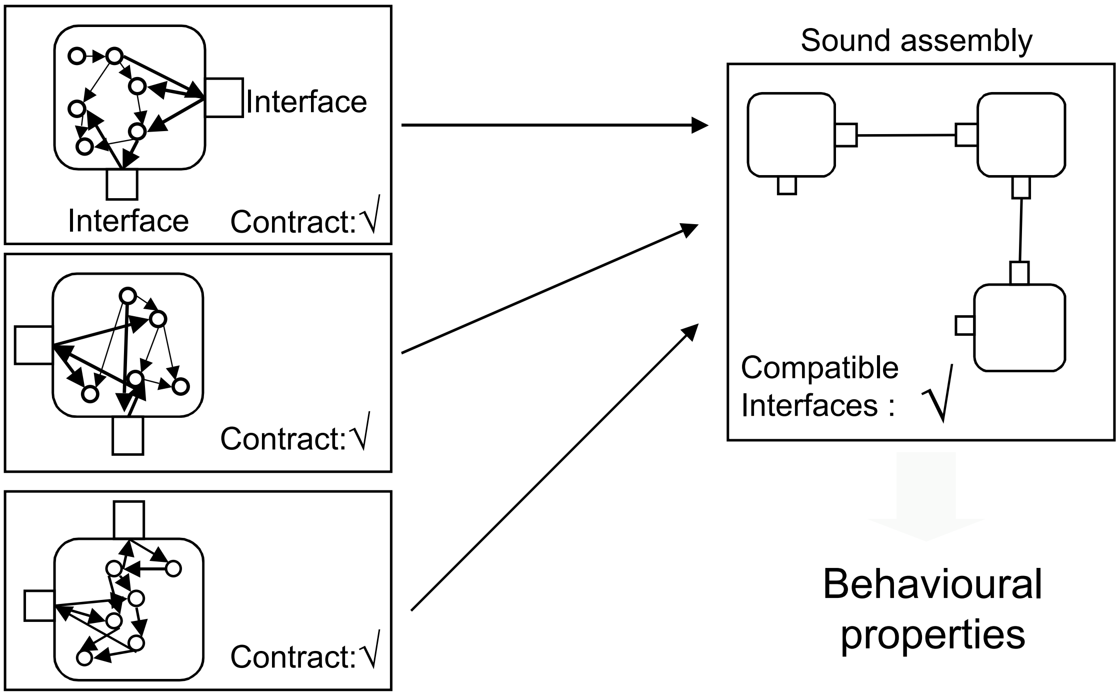


Analysis
of the global
behaviour

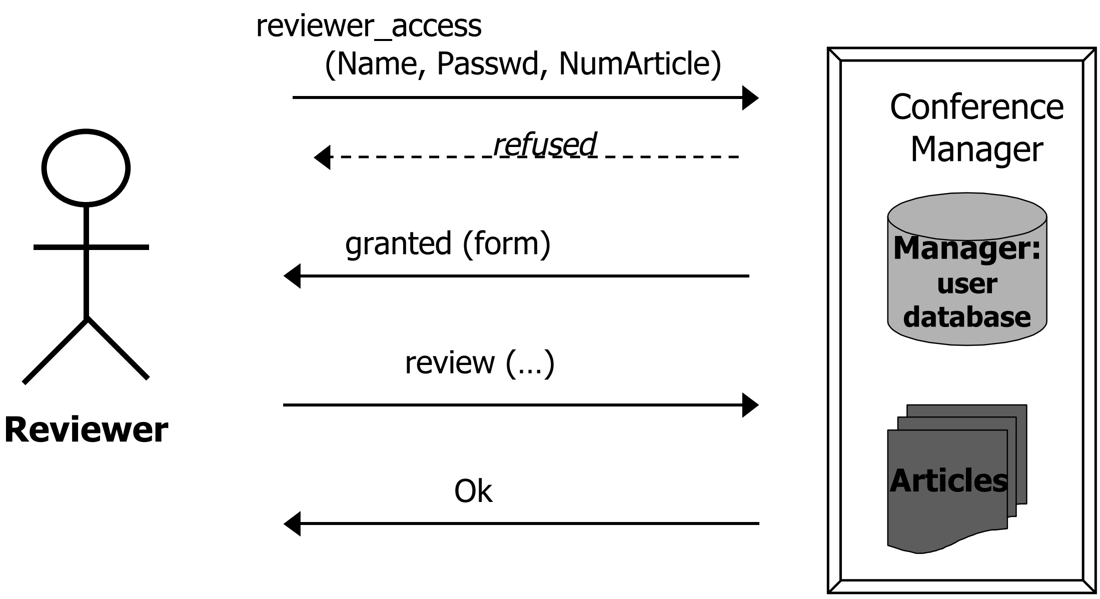
Our approach



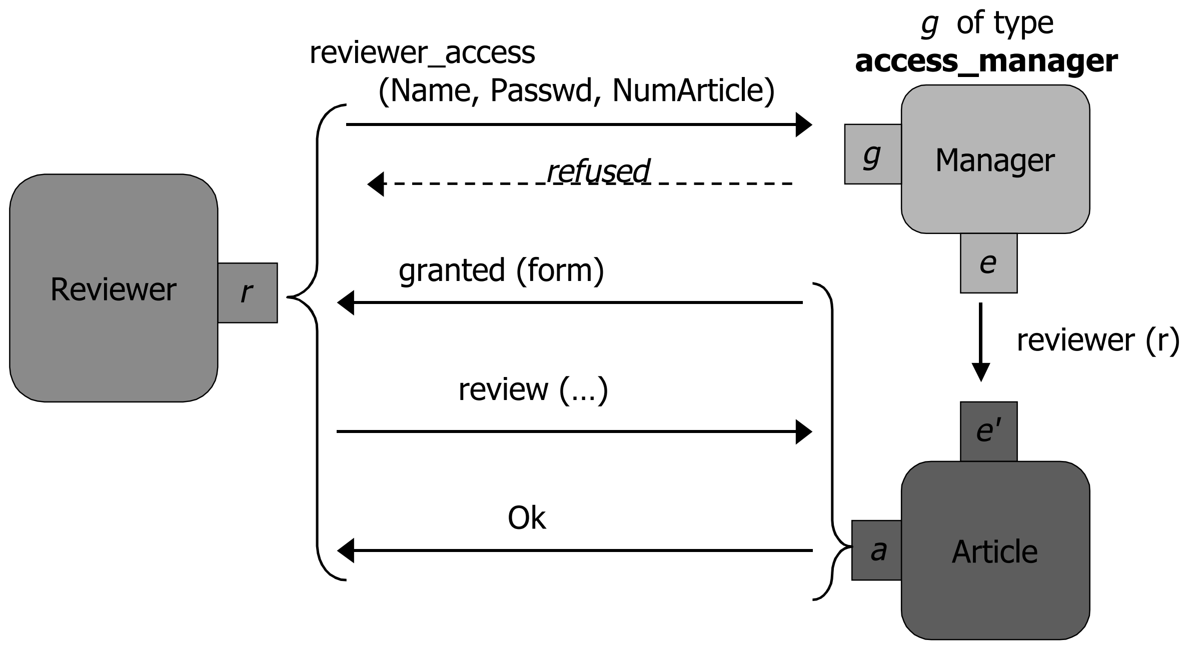
Our approach



Interface types: example



Interface types: example



Example:
Type access_manager

- **access_manager** =
  may ? [reviewer_access (string,string,integer)**;**
  must ! [refused**; 0**
  + granted (strings)**; reviewer_manager]]**

- **reviewer_manager** =
  must ? [review (strings)**; must ! [** Ok**; reviewer_manager_chg**
  + error**; reviewer_manager]]**
- **reviewer_manager_chg** =
  may ? [review (strings)**; must ! [** Ok**; reviewer_manager_chg**
  + error**; reviewer_manager_chg]]**

Example: Type access_manager

allowed: you can send, I guarantee the reception

- **access_manager** =
  may ? [reviewer_access (string,string,integer)**;**
  must ! [refused**; 0**
  + granted (strings)**; reviewer_manager]]**

obligation: I must send

You must send

- **reviewer_manager** =
  must ? [review (strings)**; must ! [** Ok**; reviewer_manager_chg**
  + error**; reviewer_manager]]**
- **reviewer_manager_chg** =
  may ? [review (strings)**; must ! [** Ok**; reviewer_manager_chg**
  + error**; reviewer_manager_chg]]**

Compatibility: *Comp (I, J)*

J \ *I*	**must ?**	**may ?**	**must !**	**may !**	**0**
must ?			√		
may ?		√	√	√	√
must !	√	√			
may !		√			
0		√			√

$$Comp(\ mod_I\ \mathbf{!}\ [\ \Sigma_k\ M_k\ ;\ I_k\ ],\ mod_J\ \mathbf{?}\ [\ \Sigma_l\ \ M_l\ ;\ J_l\ ]\ ) =_{\mathrm{def}}$$

$$Comp_{\mathrm{mod}}(\ mod_I\ \mathbf{!},\ mod_J\ \mathbf{?}\ )$$
$$\wedge\ (\ \forall k,\ \exists l : Comp_{\mathrm{msg}}(\ M_k,\ M_l\ ) \wedge Comp(\ I_k,\ J_l\ )\ )$$

$$Comp_{\mathrm{msg}}(\ M_{\mathbf{!}}\ (I_i\ ),\ M_{\mathbf{?}}(J_i\ )\ )\ =_{\mathrm{def}}\ M_{\mathbf{!}} = M_{\mathbf{?}}\ \wedge\ \forall i,\ I_i \leq J_i$$

Compatibility: *Comp (I, J)*

	must ?	**may ?**	**must !**	**may !**	**0**
must ?			√		
may ?		√	√	√	√
must !	√	√			
may !		√			
0		√			√

- reviewer_manager =
  must ? [review (strings)**; must ! [** Ok**;** reviewer_manager_chg
  + error**;** reviewer_manager **]]**

  reviewer_manager_chg = **may ? […]**
- enter_review =
  must ! [review (strings)**; must ? [** Ok**; 0**
  + error**;** enter_review **]]**

Compatibility: *Comp (I, J)*

	must ?	**may ?**	**must !**	**may !**	**0**
must ?			√		
may ?		√	√	√	√
must !	√	√			
may !		√			
0		√			√

- reviewer_manager =
  must ? [review (strings)**; must ! [** Ok**;** reviewer_manager_chg
  + error**;** reviewer_manager **]]**

  reviewer_manager_chg = **may ? […]**
- enter_review =
  must ! [review (strings)**; must ? [** Ok**; 0**
  + error**;** enter_review **]]**

Compatibility: *Comp (I, J)*

	must ?	**may ?**	**must !**	**may !**	**0**
must ?			√		
may ?		√	√	√	√
must !	√	√			
may !		√			
0		**√**			√

- reviewer_manager =
  must ? [review (strings)**;** **must ! [** Ok**;** reviewer_manager_chg
  + error**;** reviewer_manager **]]**

  reviewer_manager_chg = **may ? [...]**
- enter_review =
  must ! [review (strings)**; must ? [** Ok**; 0**
  + error**;** enter_review **]]**

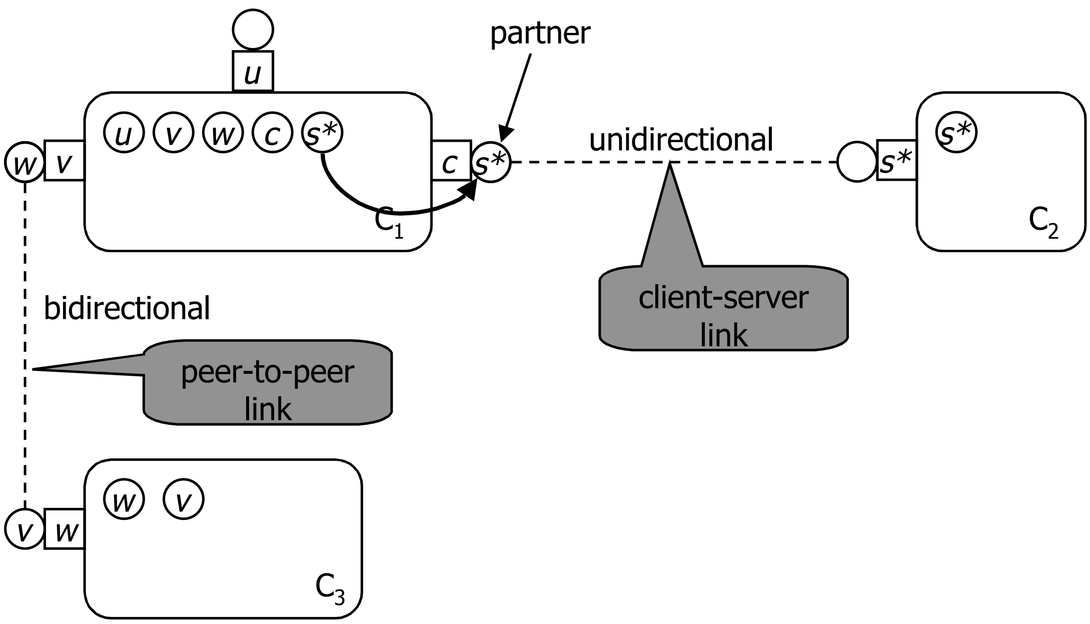
Subtyping: *T ≤ S*

- Compatibility: sent message ≤ received message
- receivings:
  – *mod* **?** $M_1+M_2+M_3 \le$ *mod* **?** M_1+M_2
  – contra-variant: $M(I) \le M(J) \Leftrightarrow J \le I$
- sendings:
  – *mod* **!** $M_1 \le$ *mod* **!** M_1+M_2
  – co-variant: $M(I) \le M(J) \Leftrightarrow I \le J$
- modalities:
  – **may ? ≤ must ?** – **may ? ≤ 0** – **may ? ≤ may !**
  – **must ! ≤ may !** – **0 ≤ may !**

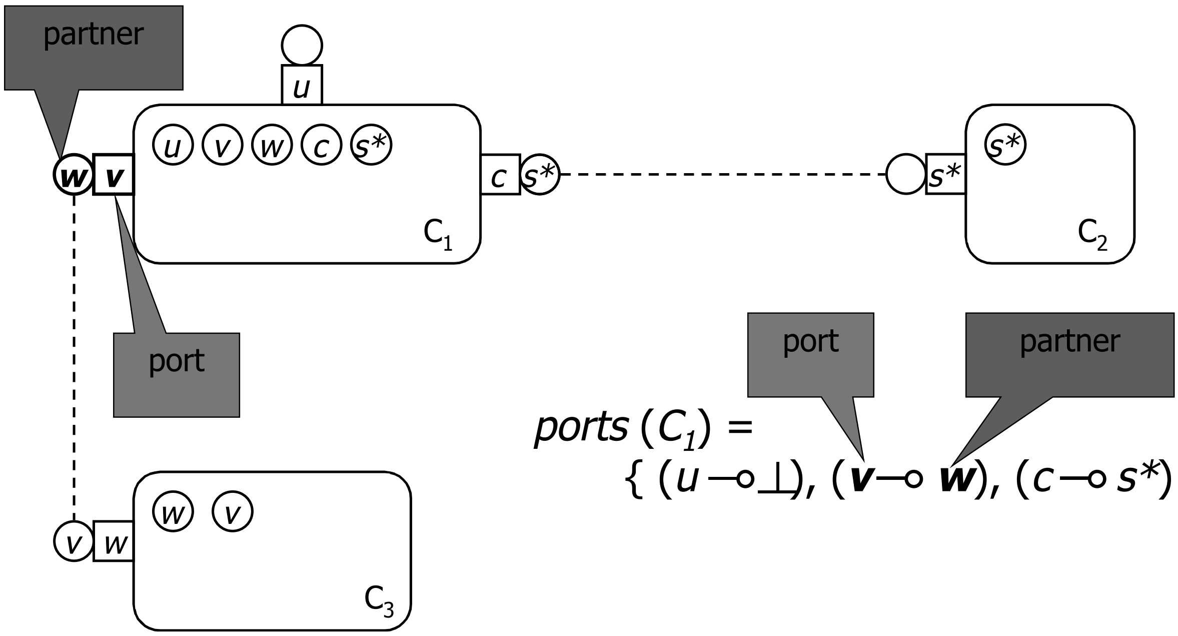
Properties of the subtypes

- $\leq$ is transitive
- Subtype can replace super-type
  - *Comp* (I, S) & $(T \leq S) \quad \Rightarrow \quad$ *Comp* $(I; T)$
- Greater compatible super-type:
  - dual: $J^{D} =_{def} J$ with reversed sendings and receivings
  - *Comp* $(I, J) \Leftrightarrow I \leq J^{D}$
- Demonstrations
  - by induction on the structure of the types

Component model

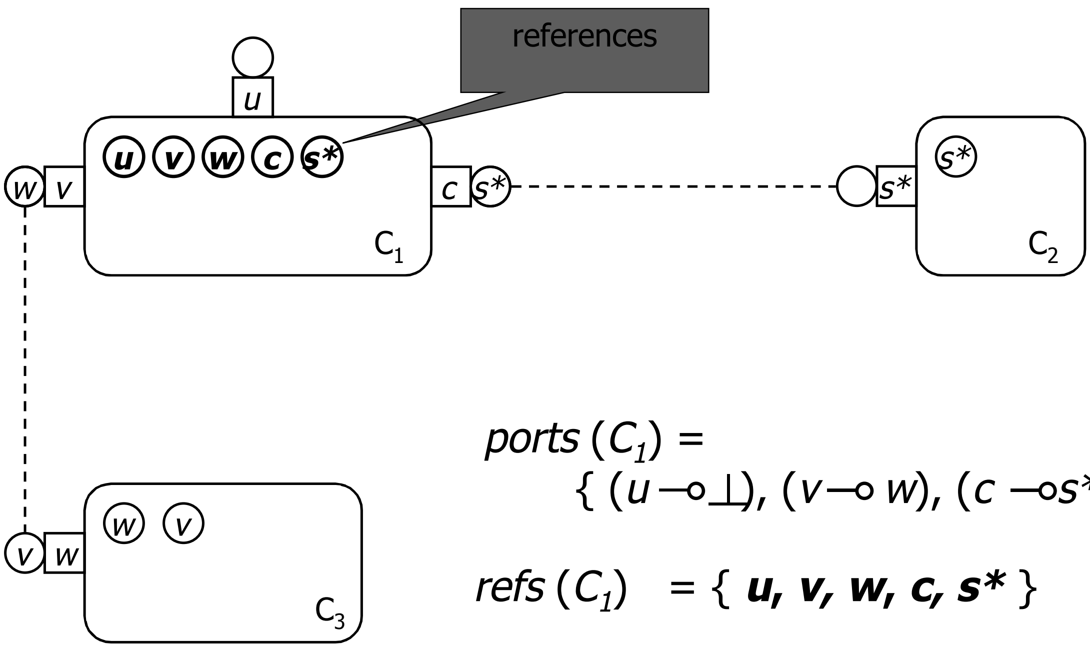


Component model



ports (C_1) =
$\{ (u \multimap \perp), (\boldsymbol{v} \multimap \boldsymbol{w}), (c \multimap s^*) \}$

Component model



ports (C_1) =
$\{ (u \multimap \perp), (v \multimap w), (c \multimap s^*) \}$

refs (C_1) = $\{$ ***u, v, w, c, s**** $\}$

Component model: ports

- Model based on observation of ports
- State of a port : $u\rho^{\sigma}$
  - ρ = action = $\begin{cases} \textbf{!} & u \text{ is in a sending state} \\ \textbf{?} & u \text{ is in a receiving state} \\ \textbf{0} & u \text{ has no action} \end{cases}$
  - σ = activity = $\begin{cases} \textbf{a} & u \text{ is active} \\ \textbf{s} & u \text{ is suspended} \\ \textbf{i} & u \text{ is idle} \end{cases}$
- Example:
  - $u\ ?^{\mathbf{a}}$ = active in receiving $u\ !^{\mathbf{s}}$ = suspended in sending

Component model: threads

- Multi-threaded components
- Dependencies between ports: $x \rightarrowtail y$
  - activity of x is suspended until y terminates or becomes idle
- A thread is a chain *(head, queue)*
  - *head:* current active port,
  - *queue:* ordered sequence of suspended ports
  - can dynamically grow/diminish

$$\underbrace{u_1\mathbf{!^s} \rightarrowtail u_2\mathbf{!^s} \rightarrowtail \ldots\ldots u_{n-1}\mathbf{!^s}}_{\text{queue}} \rightarrowtail \underbrace{u_n\mathbf{?^a}}_{\text{head}}$$

Component model: threads

- Multi-threaded components
- Dependencies between ports: $x \rightarrowtail y$
  - activity of *x* is suspended until *y* terminates or becomes idle
- A thread is a chain *(head, queue)*
  - *head:* current active port,
  - *queue:* ordered sequence of suspended ports
  - can dynamically grow/diminish

$$\underbrace{u_1\mathbf{!}^{\mathbf{s}} \rightarrowtail u_2\mathbf{!}^{\mathbf{s}} \rightarrowtail \ldots\ldots}_{\text{queue}}\ \underbrace{u_{n-1}\mathbf{!}^{\mathbf{a}}}_{\text{head}}$$

Component semantic

- Component: B (P, R, T)

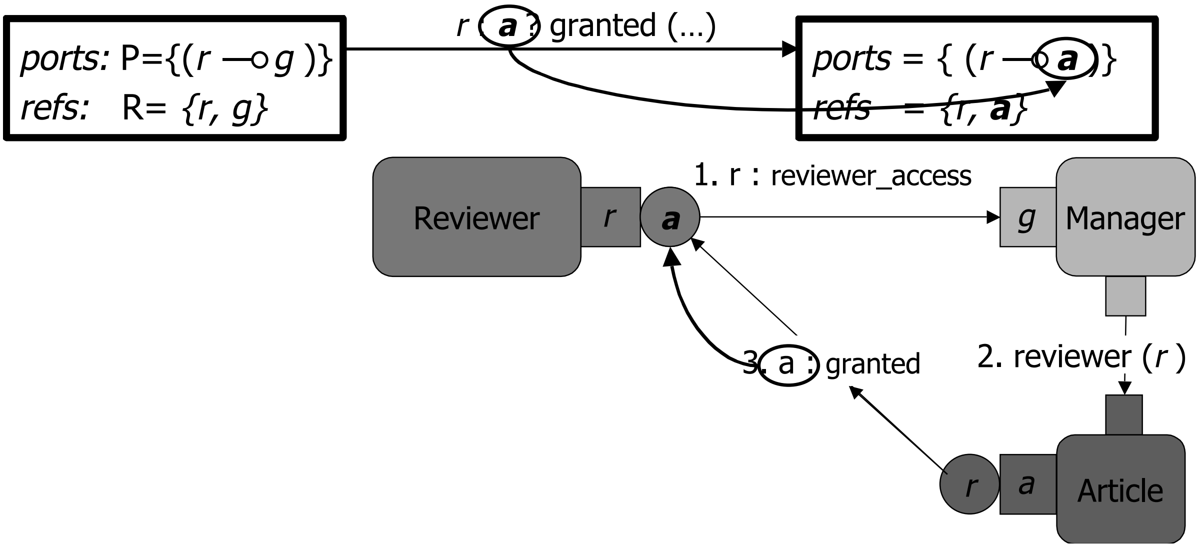
  state ports, references, threads

- Operational semantic

  async. com.: Fifo queues

  - *B* (P, R, T), *Com* $\longrightarrow$ *B'* (P', R', T'), *Com'*
- 11 Rules:
  - creation / removal of ports
  - binding
  - (de)activation of ports (idle, active, suspended)
  - sending/receiving messages

Example: RECV for Reviewer component



$$\frac{\mathsf{T}' = \mathsf{T}[u\rho/u\mathbf{?}] \qquad \mathsf{R}' = \mathsf{R} \cup \{\mathrm{refs}(\tilde{v}), u''\} - \{u' | (u \multimap u') \wedge \mathrm{peer}(u')\} \qquad Com' = Com[u\triangleright] \qquad \boxed{\mathsf{P}' = \mathsf{P}[u \multimap u''] \text{ si } \mathrm{peer}(u)}}{B(\mathsf{P},\mathsf{R},\mathsf{T}), Com \xrightarrow{u:u''\mathbf{?}M(\tilde{v})} B'(\mathsf{P}',\mathsf{R}',\mathsf{T}'), Com'} \ \triangledown$$

Some other rules

$$\text{C-BIND}\ \frac{\mathsf{P}' = \mathsf{P}[u \multimap v]}{B(\mathsf{P},\mathsf{R},\mathsf{T}), Com \xrightarrow{\mathrm{bind}(u \multimap v)} B'(\mathsf{P}',\mathsf{R},\mathsf{T}), Com}\ \square$$

$$\square \triangleq (u \multimap \bot) \ \wedge \boxed{\mathsf{T}(u) = !^{\mathsf{a},\mathsf{i}}} \wedge \ v \in \mathsf{R} \ \wedge \boxed{(\mathrm{peer}(v) \Rightarrow v \notin CoDom(\mathsf{P}))}$$

-only sending ports, not suspended
-peer reference is attached to 1 port

$$\text{C-ACTV}\ \frac{\mathsf{T}' = \mathsf{T}[u \rightarrowtail v]}{B(\mathsf{P},\mathsf{R},\mathsf{T}), Com \xrightarrow{\mathrm{actv}(u \rightarrowtail v)} B'(\mathsf{P},\mathsf{R},\mathsf{T}'), Com}\ \mathsf{T}(u) = !^{\mathsf{a}} \ \wedge \boxed{\mathsf{T}(v) = !^{\mathsf{i}}}$$

-a port cannot suspend on a receiving port

$$\text{C-SEND}\ \frac{\boxed{\mathsf{R}' = \mathsf{R} - \mathrm{peer}(\tilde{v} \cup \{u\})} \qquad \mathsf{T}' = \mathsf{T}[u\rho/u!] \qquad Com' = Com[u' \triangleleft u : M(\tilde{v})]}{B(\mathsf{P},\mathsf{R},\mathsf{T}), Com \xrightarrow{u:u'!M(\tilde{v})} B'(\mathsf{P},\mathsf{R}',\mathsf{T}'), Com'}\ \triangle$$

-peer reference is private: known only to the partner

Component and contracts

- Contractual component: $B(\ldots),\tilde{C}$
  - correct behaviour

$$\frac{\tilde{C} \xrightarrow{\alpha} \tilde{C}' \qquad B(\ldots) \xrightarrow{a} B'(\ldots) \qquad a:\alpha}{B(\ldots),\tilde{C} \xrightarrow{a:\alpha} B(\ldots),\tilde{C}'}$$

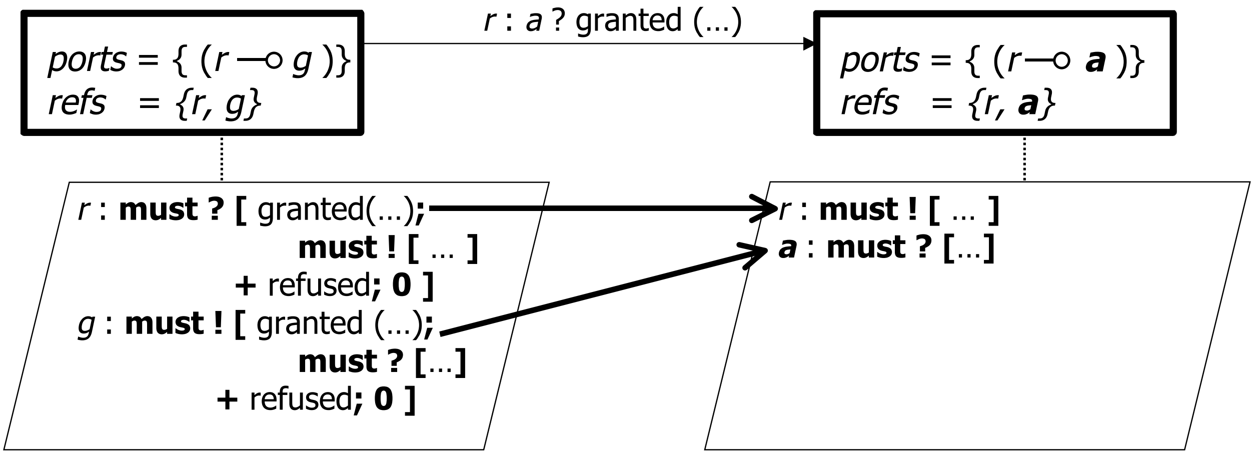
  - unauthorized transition

$$\frac{\tilde{C} \not\xrightarrow{\alpha} \tilde{C}' \qquad B(\ldots) \xrightarrow{a} B'(\ldots) \qquad a:\alpha}{B(\ldots),\tilde{C} \xrightarrow{a:\alpha} Error}$$

  - missing required transition

$$\frac{\tilde{C} \xrightarrow{\alpha} \tilde{C}' \qquad B(\ldots) \not\xrightarrow{a} B'(\ldots) \qquad a:\alpha}{B(\ldots),\tilde{C} \xrightarrow{a:\alpha} Error} \quad \mathrm{mod}\,(\alpha) = \textbf{must}$$

Example: RECV for Reviewer component



$$\frac{u{:}T \equiv mod\ \textbf{?}\ M_\Sigma \qquad u'{:}T' \equiv mod'\ \textbf{!}\ M'_\Sigma \qquad B(\mathsf{P},\mathsf{R},\mathsf{T}) \xrightarrow{u:u''\textbf{?}m_k} B'(\mathsf{P}',\mathsf{R}',\mathsf{T}')}{(B(\mathsf{P},\mathsf{R},\mathsf{T}),\tilde{C}) \xrightarrow{u:u''\textbf{?}m_k} (B'(\mathsf{P}',\mathsf{R}',\mathsf{T}'),\boxed{\tilde{C}[u{:}T_k/T,}\boxed{u''{:}T'_k/u'{:}T']} \Leftarrow \tilde{v}'{:}\tilde{U}'_k)}$$

Some other rules

$$\text{BIND}\ \frac{u\!:\!T \qquad v\!:\!S \qquad B(\mathsf{P},\mathsf{R},\mathsf{T}) \xrightarrow{\text{bind}(u \multimap v)} B'(\mathsf{P}',\mathsf{R},\mathsf{T})}{(B(\mathsf{P},\mathsf{R},\mathsf{T}),\tilde{C}) \xrightarrow{\text{bind}(u \multimap v)} (B'(\mathsf{P}',\mathsf{R},\mathsf{T}),\tilde{C})}\ Comp(T,S)$$

$$\text{BIND-ERR}\ \frac{u\!:\!T \qquad v\!:\!S \qquad B(\mathsf{P},\mathsf{R},\mathsf{T}) \xrightarrow{\text{bind}(u \multimap v)} B'(\mathsf{P}',\mathsf{R},\mathsf{T})}{(B(\mathsf{P},\mathsf{R},\mathsf{T}),\tilde{C}) \rightarrow \mathit{Error}}\ \neg Comp(T,S)$$

$$\text{RECV-ERR}\ \frac{u\!:\!T \equiv mod\ ?[*]M_{\Sigma} \qquad \forall k, B(\mathsf{P},\mathsf{R},\mathsf{T}) \not\xrightarrow{u:u'?m_k} B'(\mathsf{P}',\mathsf{R}',\mathsf{T}')}{(B(\mathsf{P},\mathsf{R},\mathsf{T}),\tilde{C}) \rightarrow \mathit{Error}}$$

$$\text{RECV-UN}\ \frac{u\!:\!T \equiv mod\ ?\ M_{\Sigma} \qquad B(\mathsf{P},\mathsf{R},\mathsf{T}) \xrightarrow{u:u'?m_k} B'(\mathsf{P}',\mathsf{R}',\mathsf{T}')}{(B(\mathsf{P},\mathsf{R},\mathsf{T}),\tilde{C}) \xrightarrow{u:u'?m_k} (B'(\mathsf{P}',\mathsf{R}',\mathsf{T}'),\tilde{C}[u\!:\!T_k/T] \Leftarrow \boxed{u'\!:\!T_k{}^{\mathcal{D}},}\tilde{v}\!:\!\tilde{U}_k)}\ \blacktriangle \wedge (u \multimap \bot)$$

-RECV from unknown partner: take the greater type

Sound assembly of components

- Component honouring a contract
  - *B* is well-typed: *B(P,R,T),*$\tilde{C}$ never leads to *Error*
- Assembly of components:

$$\mathcal{A} = \{\ (B_1(\mathsf{P}_1,\mathsf{R}_1,\mathsf{T}_1),\tilde{C}_1), \ldots, (B_n(\mathsf{P}_n,\mathsf{R}_n,\mathsf{T}_n),\tilde{C}_n), Com\ \}$$

  - reference closed
  - only client/server and peer-to-peer bindings
  - all ports are active and independent
- Sound assembly:
  - all components respect their contract
  - ports bound to each other are compatible

Properties

Soundness is maintained through evolution

– a sound configuration of components never leads to *Error*

$$\forall\, C : \mathcal{A} \longrightarrow^{*} C, \quad C \not\longrightarrow Error$$

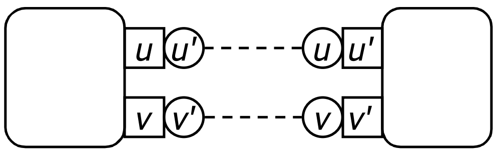
All the messages are eventually consumed

$$\forall\, u,v,i,M :\ (u \multimap v) \in \mathsf{P}_i\,,\ C \xrightarrow{u:v\,!\,M} C'$$

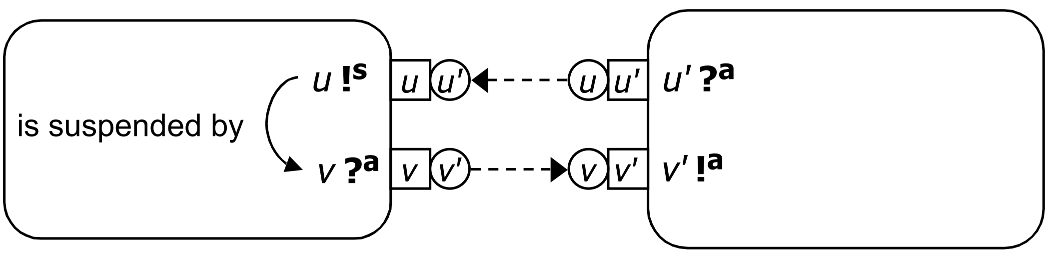
$$\Rightarrow \exists\, C'',C''' \text{ such that } C' \longrightarrow^{*} C'' \xrightarrow{v:u\,?\,M} C'''$$

External deadlock

- During assembly : no verification of the global behaviour
  - *u* and *u'* types are compatible
  - *v* and *v'* types are compatible

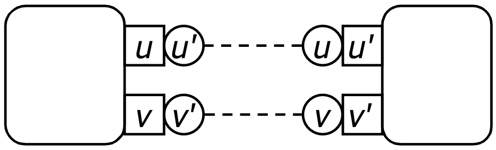


- During execution :

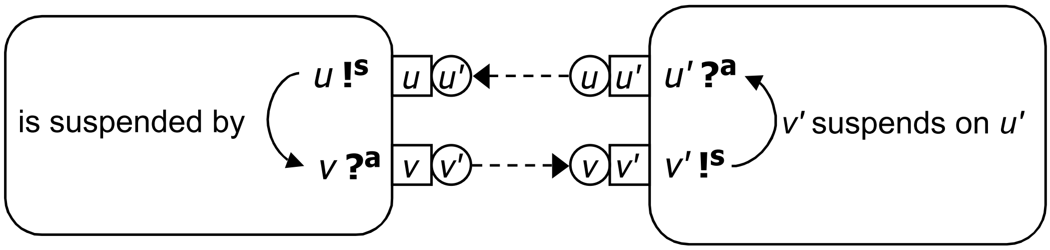


External deadlock

- During assembly : no verification of the global behaviour
  - *u* and *u′* types are compatible
  - *v* and *v′* types are compatible



- During execution :



Property: external deadlock freeness

- A port cannot suspend on a receiving port

> – external deadlock:
>
> – $u\ S\ v =_{\text{def}} u \rightarrowtail v \ \vee\ u \dashrightarrow v$ ($\dashrightarrow$ external dependency)
>
> – Ext_deadlock (C) $=_{\text{def}}$
>
> $\exists\ (u_i)_{1..n} \in C$ such that $\forall\ k < n :\ \ u_i\ S\ u_{i+1}\ \wedge\ u_n\ S\ u_1$

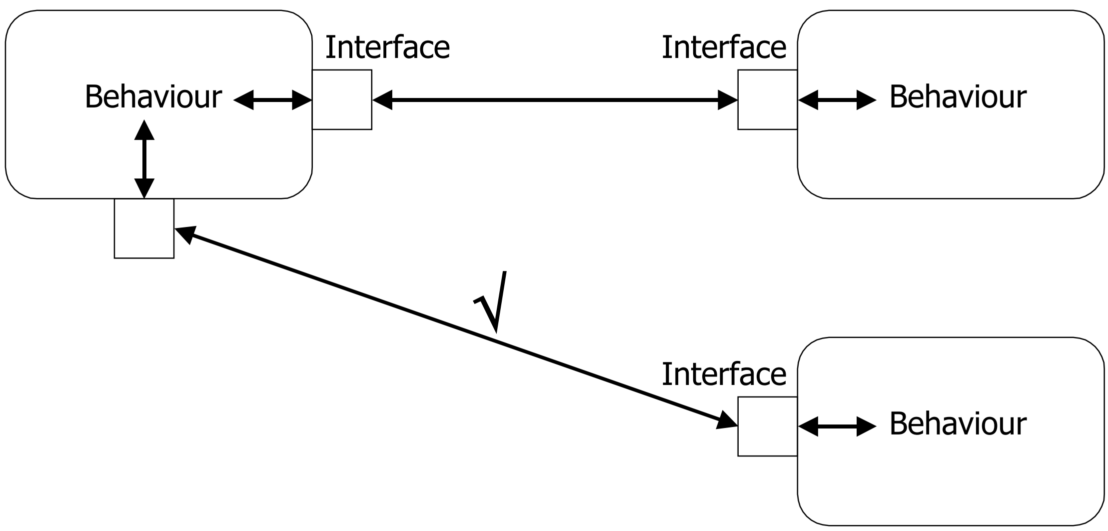
- Demonstration (deadlock freeness):
  - by induction & Reductio ad absurdum

Constraints on the component

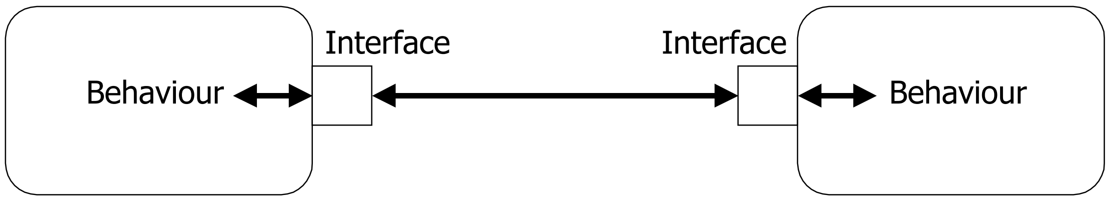
- a port cannot suspend on a receiving port:
  - $u!^{a} \xrightarrow{\text{actv}(u \rightarrowtail v)} u!^{s} \rightarrowtail v?^{a}$ is not allowed
  - $u!^{a} \xrightarrow{\text{actv}(u \rightarrowtail v)} u!^{s} \rightarrowtail v!^{a} \xrightarrow{v\text{:..!M(..)}} u!^{s} \rightarrowtail v?^{a}$ is allowed
- a receiving port cannot be suspended: $u?^{s}$ forbidden
- bindings: only sending & (active or idle) ports: $u!^{a,i}$
- a '**must !**' is not suspended by a '**may ?**'
- unbind is not allowed
- [nonrentrant servers]

Application

- Sound extension of running application



Conclusion

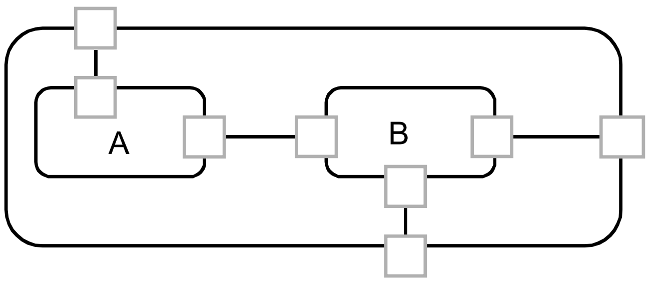


- Contract conformance: ↔ verification during compilation
- Compatible interfaces: ↔ verification during deployment

- Properties of a sound assembly
  - safety: a configuration never leads to *Error*
  - safety: external deadlock freeness
  - liveness: all sent message are eventually consumed

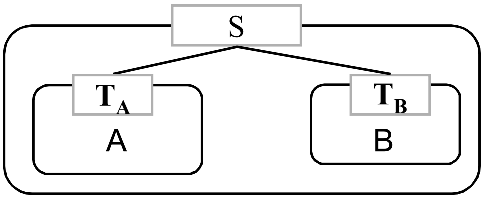
Perspectives

- Interfaces: infinite state machines
- Integration to existing component platforms
- UML Profile
- Composite components & delegation:



Future Work

- Application to UML2.0: multiple delegation



- Application to a language
- From interface contracts to component contracts
- Extension to timed interfaces

- Application to PATS!!

Get the slides!
www.cyril-carrez.net
www.item.ntnu.no/~carrez