Web Service Interfaces

Dirk Beyer

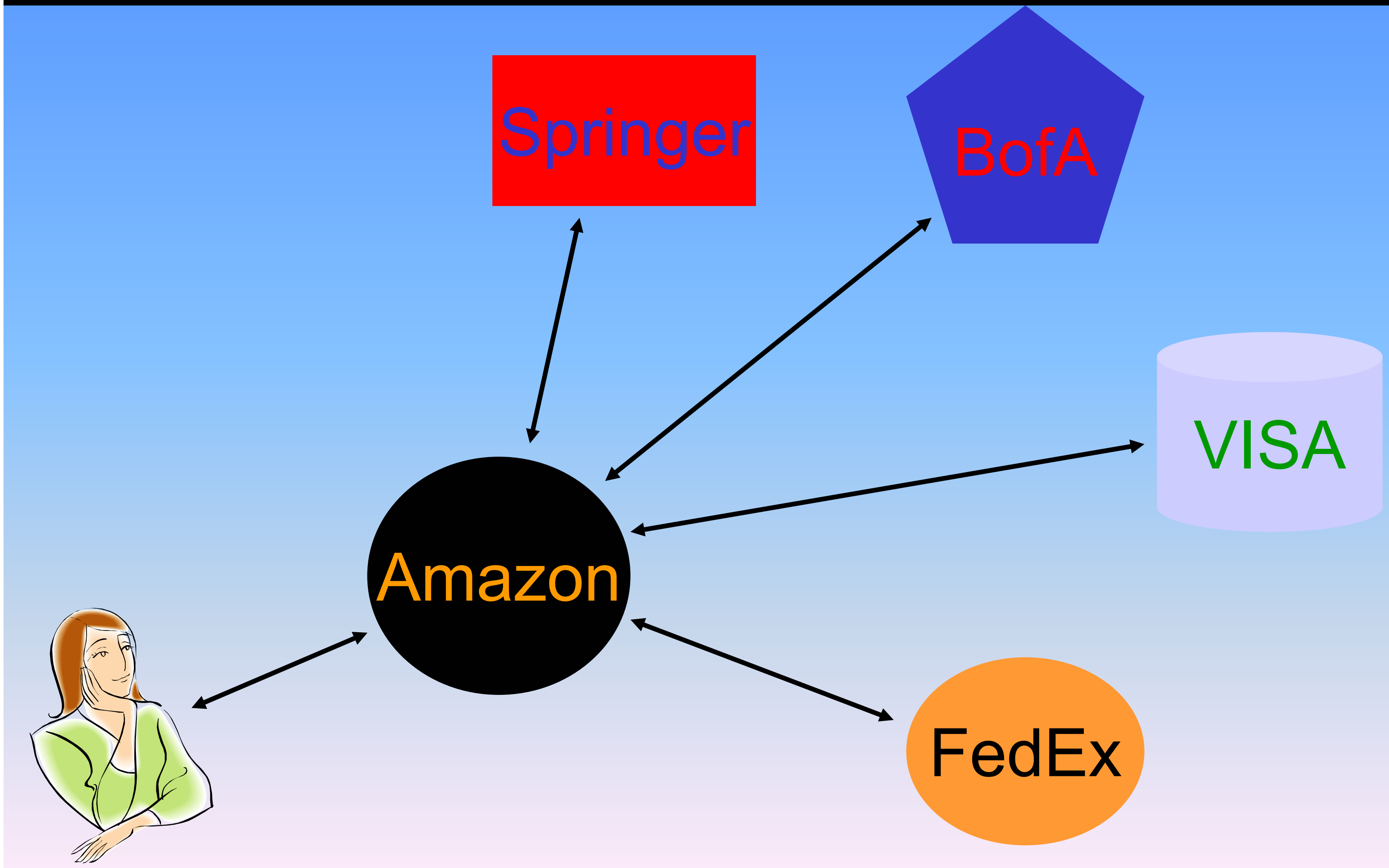
Arindam Chakrabarti

Thomas A. Henzinger

Berkeley
University of California

EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Buying a Book


Springer

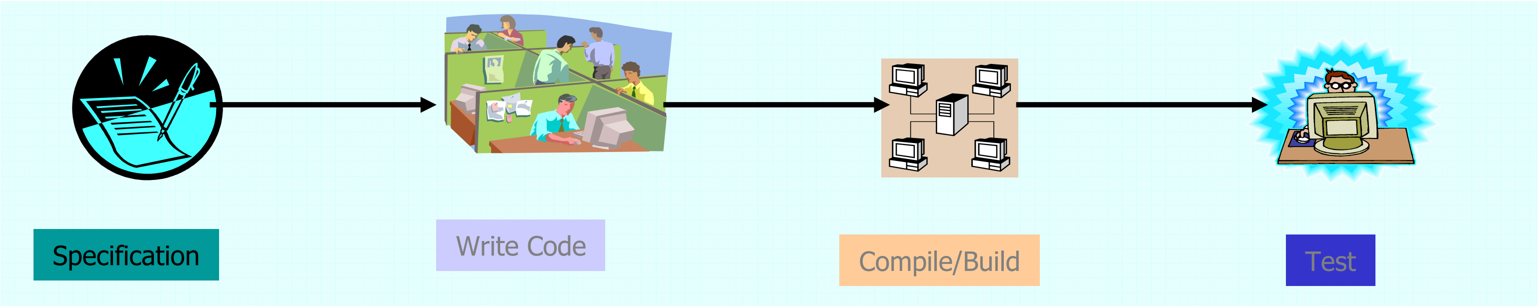
BofA

VISA

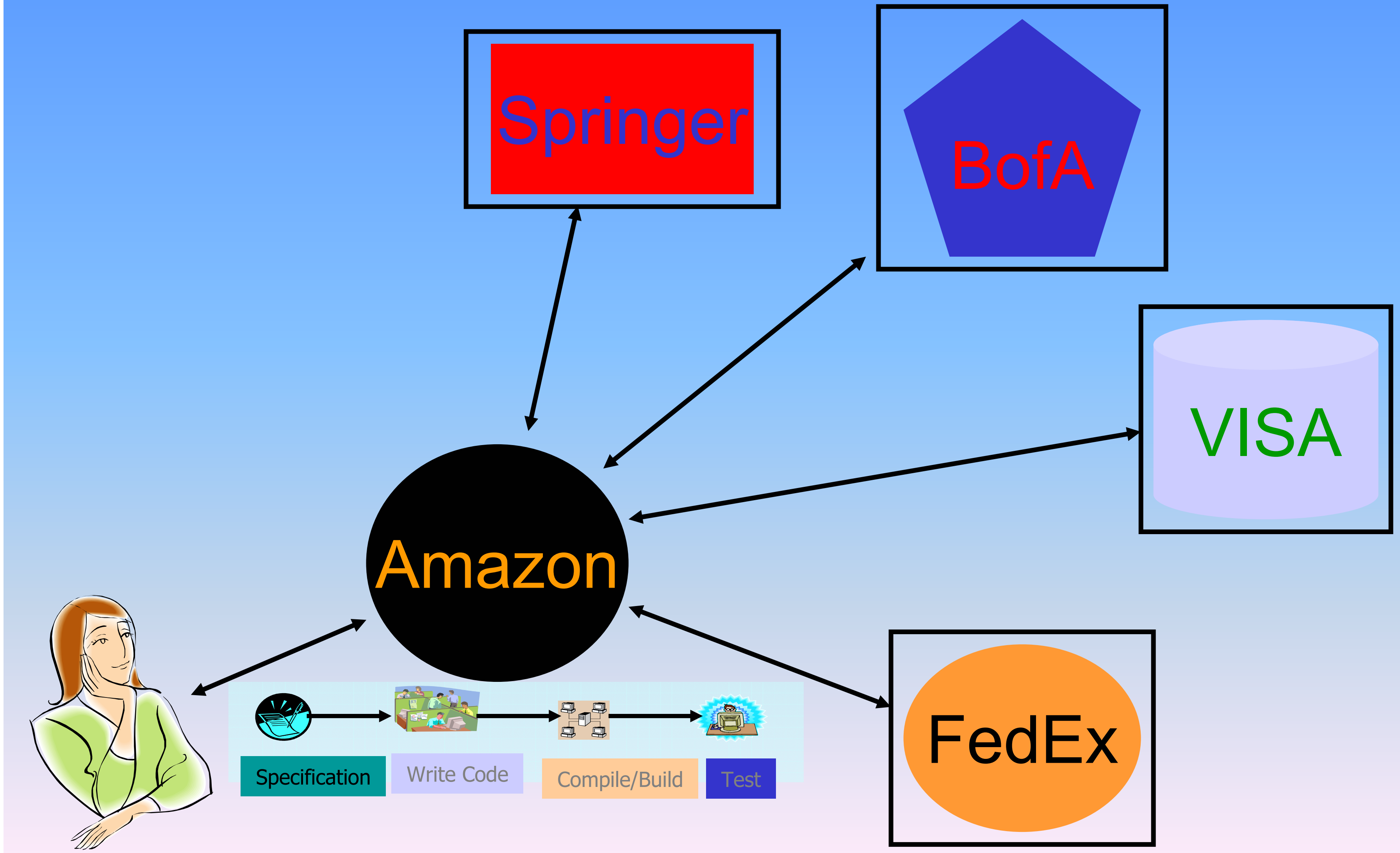
Amazon

FedEx

Stand-alone software development scenario



Web software development scenario



Springer

BofA

VISA

Amazon

FedEx

Specification

Write Code

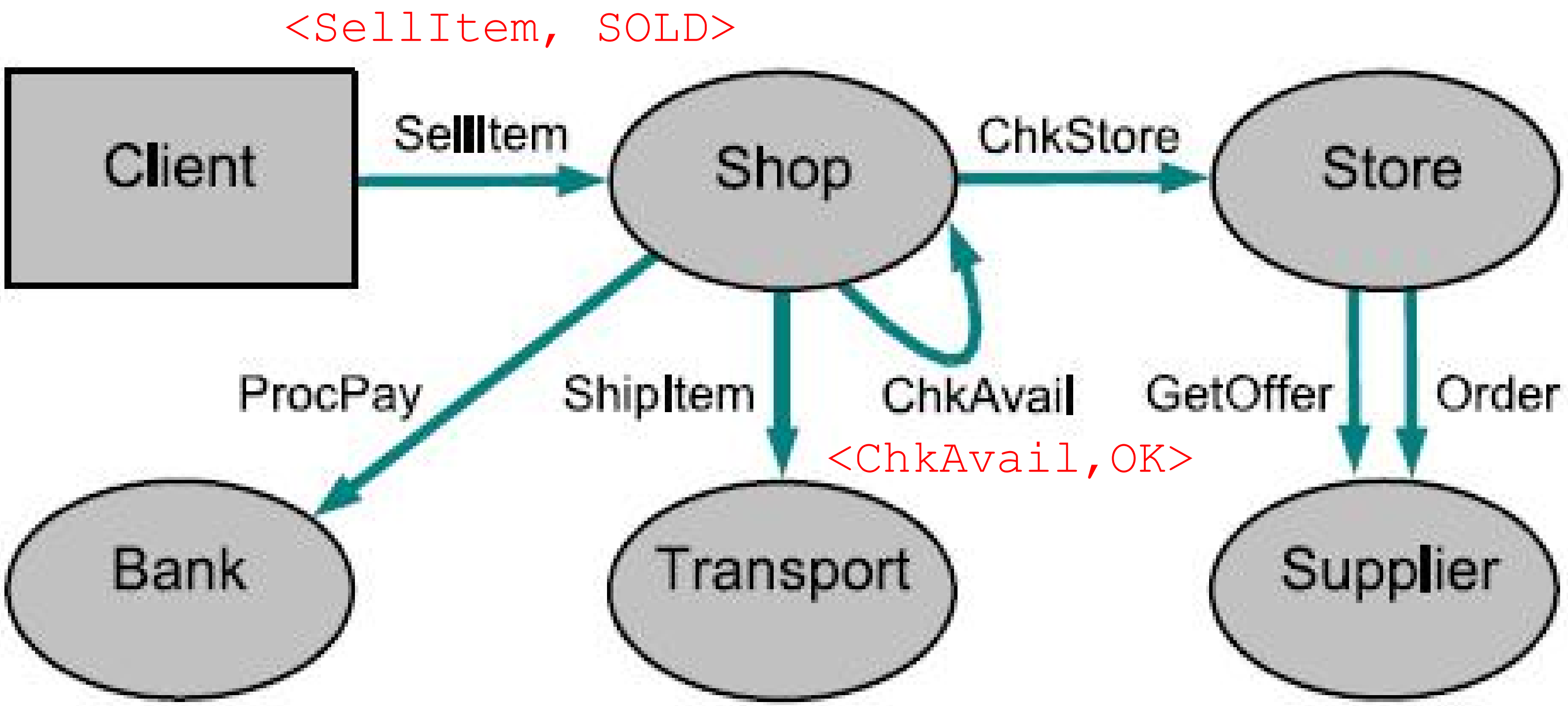
Compile/Build

Test

Web Service Interfaces

- Web methods: $m \in M$
- Namespace: $N \subseteq M$
- Instances: i
- Actions: <m,i>

Actions


<SellItem, SOLD>
Client
SellItem
Shop
ChkStore
Store
ProcPay
ShipItem
ChkAvail
GetOffer
Order
<ChkAvail,OK>
Bank
Transport
Supplier

Web Service Interfaces

We provide three of them:

- Signature interfaces
- Consistency interfaces
- Protocol interfaces

Signature Interfaces

Action *a* is mapped to the set of actions that may be directly invoked when *a* occurs.

<SellItem, sold> $\rightarrow$ { <ChkAvail, ok>, <ProcPay, ok> }

Supported and Required Actions

$<m_1,i_1> \rightarrow \{ <m_2,i_2>, <m_3,i_3>\}$

$<m_1,i_1>$ is supported (guaranteed).

$<m_2,i_2>$ and $<m_3,i_3>$ are required (assumed).

Required actions outside interface namespace are supported by the *environment.*

Compatibility and Composition

Two signature interfaces are compatible if their namespaces do not clash.

The composition is obtained by taking the union (of the namespaces and mapping functions).

Refinement

$$\langle m_1,i_1\rangle \rightarrow \{ \langle m_2,i_2\rangle, \langle m_3,i_3\rangle, \langle m_4,i_4\rangle \}$$

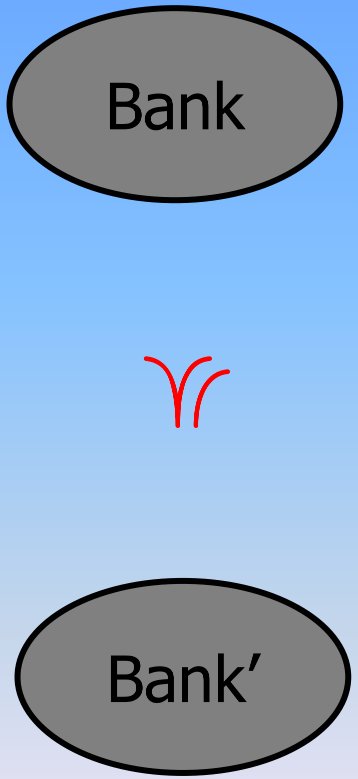
$$\curlyvee$$

$$\langle m_1,i_1\rangle \rightarrow \{ \langle m_2,i_2\rangle, \langle m_3,i_3\rangle \}$$

$$\langle m_2,i_2\rangle \rightarrow \{ \langle m_4,i_4\rangle \}$$

The refinement must not have a larger namespace.

Refinement



Bank

⋎

Bank’

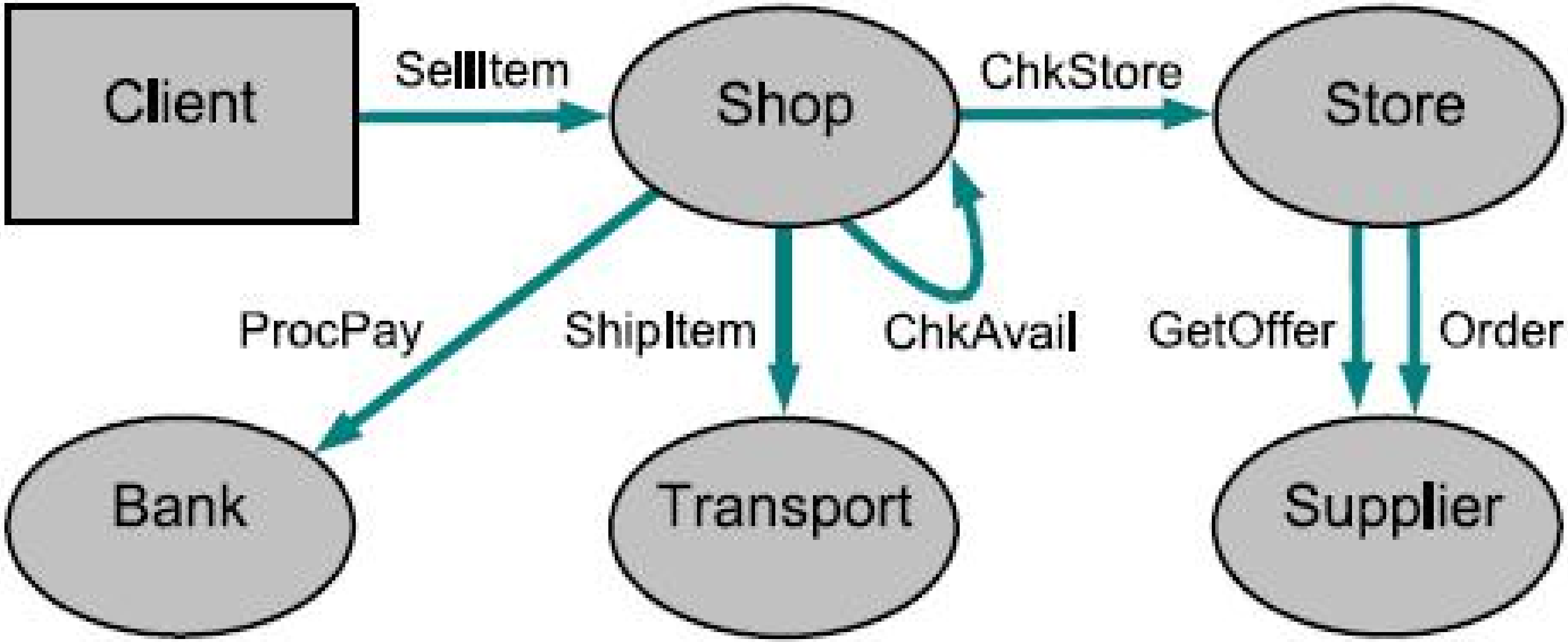
Consistency Interfaces

Consistency Interfaces

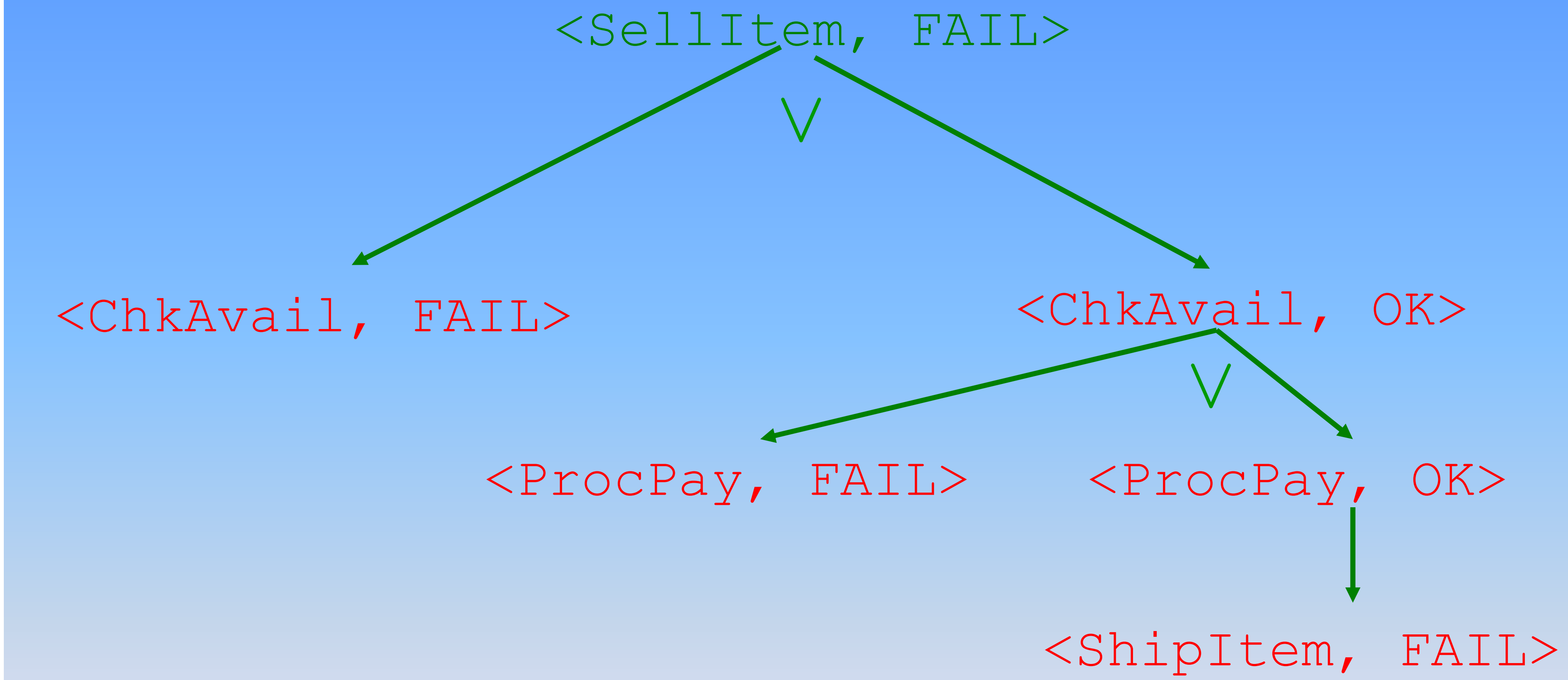
Consistency Interface =
Signature Interface + Choice

An action is mapped to a set of *conversations,* where conversation is a set of actions exhibited together (in some sequence).

Supply Chain Management Application



Supply Chain Management Application



Each path in the tree generates a conversation.

Compatibility and Composition

- Consistency interfaces have *underlying* signature interfaces.
- Two consistency interfaces are **compatible** if their underlying signature interfaces are.
- The **composition** is given by union (of namespaces and mapping functions).

Refinement

$$<m_1,i_1> \rightarrow \{\{<m_2,i_2>, <m_4,i_4>\} \vee \{<m_3,i_3>\} \}$$

$$\curlyvee$$

$$<m_1,i_1> \rightarrow \{\{<m_2,i_2>\} \vee \{<m_3,i_3>\}\}$$
$$<m_2,i_2> \rightarrow \{ \{<m_4,i_4>\} \}$$

The refinement must not have a larger namespace.

Specifications

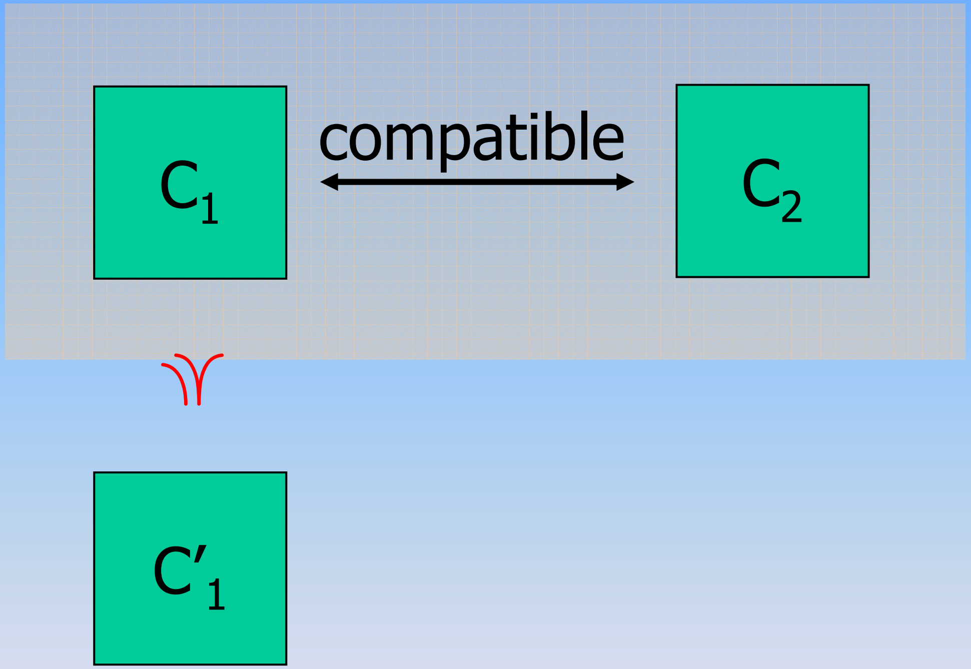
Specifications are used to capture additional behavioral properties of interest to the designer.

<SellItem,FAIL> $\not\rightsquigarrow$
{<ChkStore,FAIL>, <ProcPay,OK>}

The customer should not be charged if the item was not in store.

Compositional Refinement

If



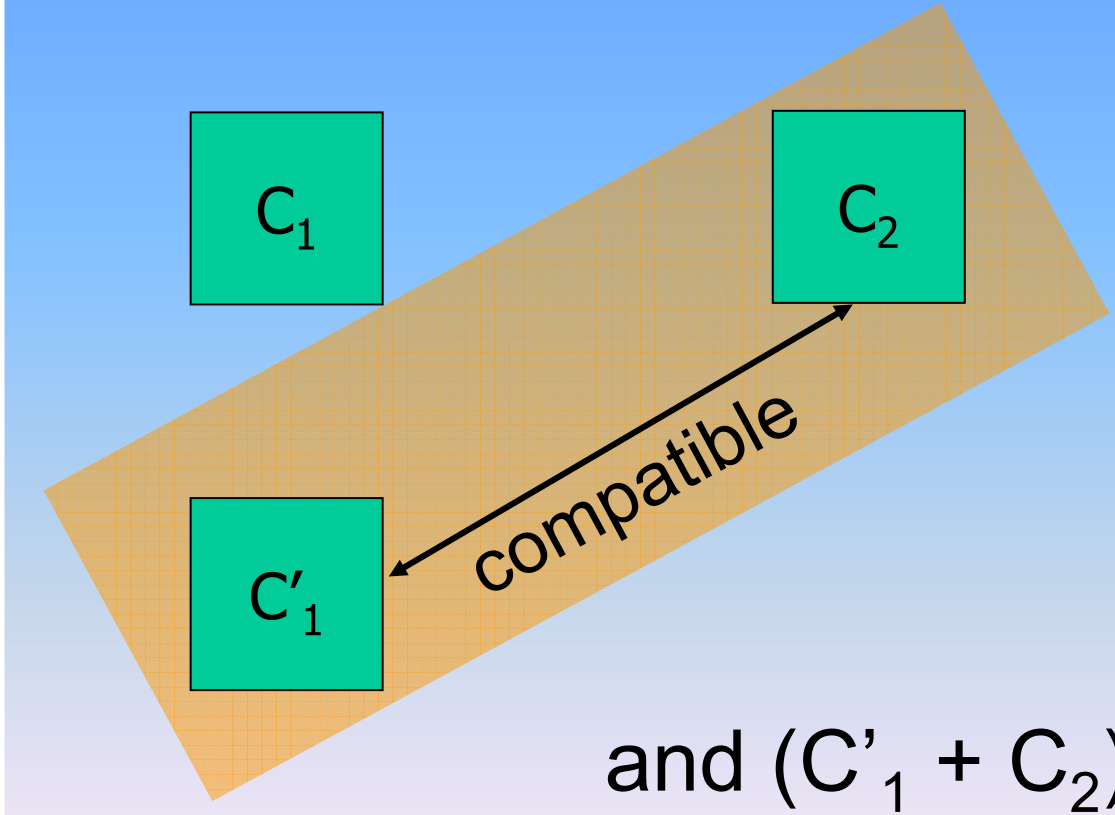
C_1 compatible C_2

$\curlyvee$

C'_1

Compositional Refinement

Then



and $(C'_1 + C_2) \preccurlyeq (C_1 + C_2)$

Refinement Preserves Specifications

If C $\models \phi$, then for all C’ such that C’ $\preccurlyeq$ C, we have C’ $\models \phi$.

Existence of Environment

$C \vDash \phi$ if and only if there exists an environment E of C, such that $(C + E) \vDash \phi$.

If C_1 and C'_1 and C_2 are such that $C'_1 \preccurlyeq C_1$, and C_1 is compatible with C_2, then for any specification ϕ if $(C_1 + C_2) \vDash \phi$ then $(C'_1 + C_2) \vDash \phi$.

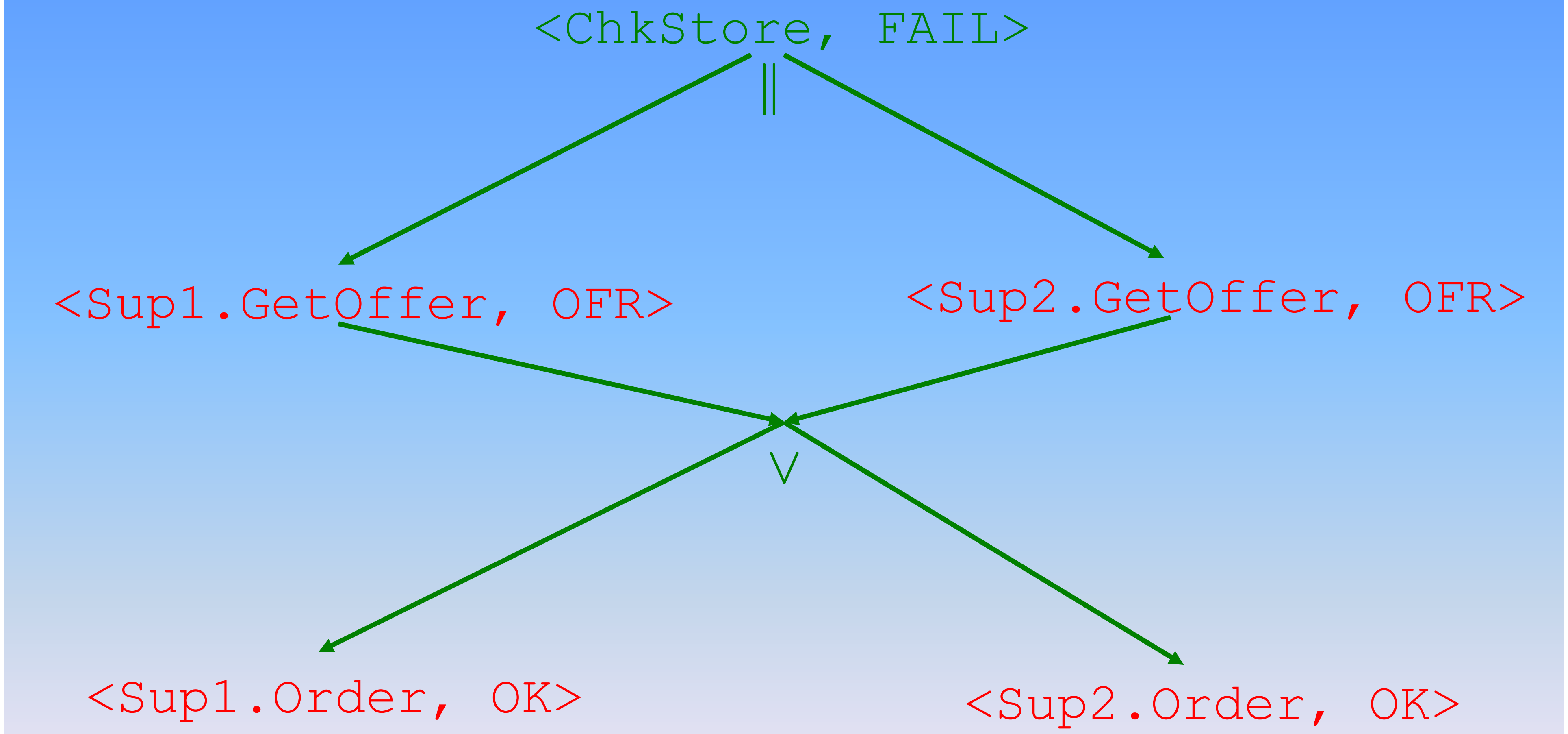
Protocol Interfaces

Protocol Interfaces

Protocol Interface = Consistency Interface + Temporal sequencing information for action invocations.

An action is mapped to a set of recursive finite state machines with transitions labeled with sets of actions.

Supply Chain Management Application



Compatibility and Composition

- Protocol interfaces have *underlying* signature interfaces.
- Two protocol interfaces are **compatible** if their underlying signature interfaces are.
- The **composition** is given by union (of namespaces and mapping functions).

Refinement

Alternating simulation

The sets of recursive finite state machines representing the protocol interfaces P and P' should be such that an alternating simulation relation holds (w.r.t. transitions labeled with actions in/not in the namespace of P/P').

Specifications

Specifications are used to capture additional properties of interest.

$$\texttt{<SellItem,FAIL>} \not\leadsto \neg\{\texttt{<ChkStore,OK>}\}\ \mathit{U}\ \{\texttt{<ProcPay,OK>}\}$$

The user should not be charged unless the item has already been found in store.

Checking Specifications

$$\frac{q_{a_0} \models (\neg C)\mathcal{W}B \ \ \ldots \ \ q_{a_k} \models (\neg C)\mathcal{W}B \quad q' \models (\neg C)\mathcal{U}B}{q \models (\neg C)\mathcal{U}B}$$

$(q, \sqcap A, q') \in \delta,$
$A \cap C = \emptyset,$
$A = \{a_0, \ldots, a_k\}$

Checking Specifications

$$\frac{q_{a_0} \models (\neg C)\mathcal{W}B \quad \ldots \quad q_{a_k} \models (\neg C)\mathcal{W}B \quad q' \models (\neg C)\mathcal{U}B}{q \models (\neg C)\mathcal{U}B} \quad \begin{array}{l}(q, \sqcap A, q') \in \delta,\\ A \cap C = \emptyset,\\ A = \{a_0, \ldots, a_k\}\end{array}$$

$$\frac{q_a \models (\neg C)\mathcal{W}B \quad q' \models (\neg C)\mathcal{W}B}{q \models (\neg C)\mathcal{W}B} \quad (q, a, q') \in \delta,\ a \notin C$$

Checking Specifications

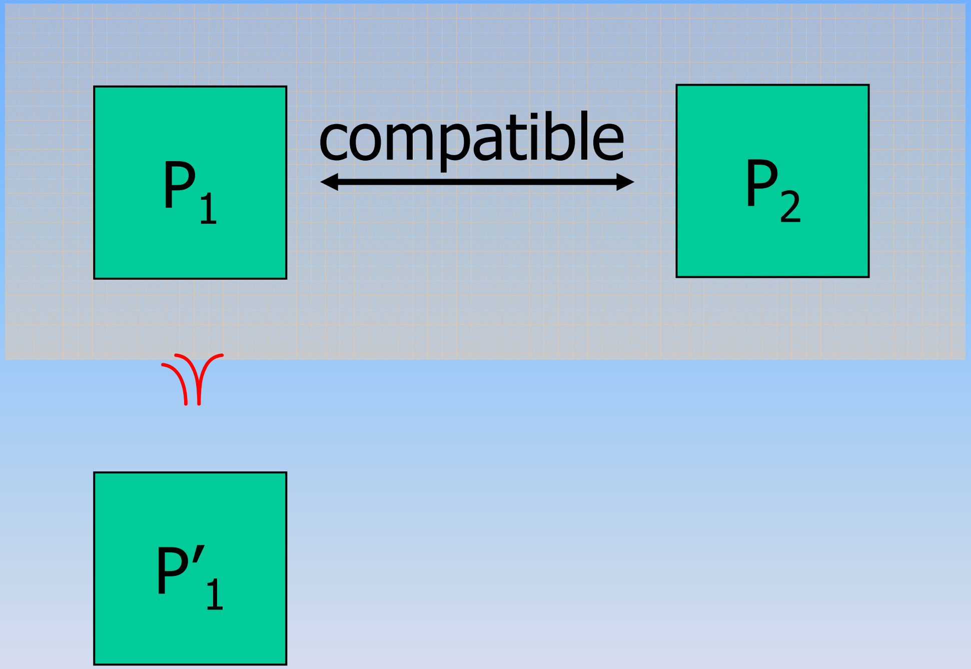
$$\frac{q_{a_0} \models (\neg C)\mathcal{W}B \;\; \dots \;\; q_{a_k} \models (\neg C)\mathcal{W}B \quad q' \models (\neg C)\mathcal{U}B}{q \models (\neg C)\mathcal{U}B} \quad \begin{array}{l}(q, \sqcap A, q') \in \delta,\\ A \cap C = \emptyset,\\ A = \{a_0, \dots, a_k\}\end{array}$$

$$\frac{q_a \models (\neg C)\mathcal{W}B \quad q' \models (\neg C)\mathcal{W}B}{q \models (\neg C)\mathcal{W}B} \quad (q, a, q') \in \delta,\; a \notin C$$

$$\frac{q_{a_0} \models (\neg C)\mathcal{W}B \;\; \dots \;\; q_{a_k} \models (\neg C)\mathcal{W}B \quad q' \models (\neg C)\mathcal{W}B}{q \models (\neg C)\mathcal{W}B} \quad \begin{array}{l}(q, \sqcap A, q') \in \delta,\\ A \cap C = \emptyset,\\ A = \{a_0, \dots, a_k\}\end{array}$$

Compositional Refinement

If



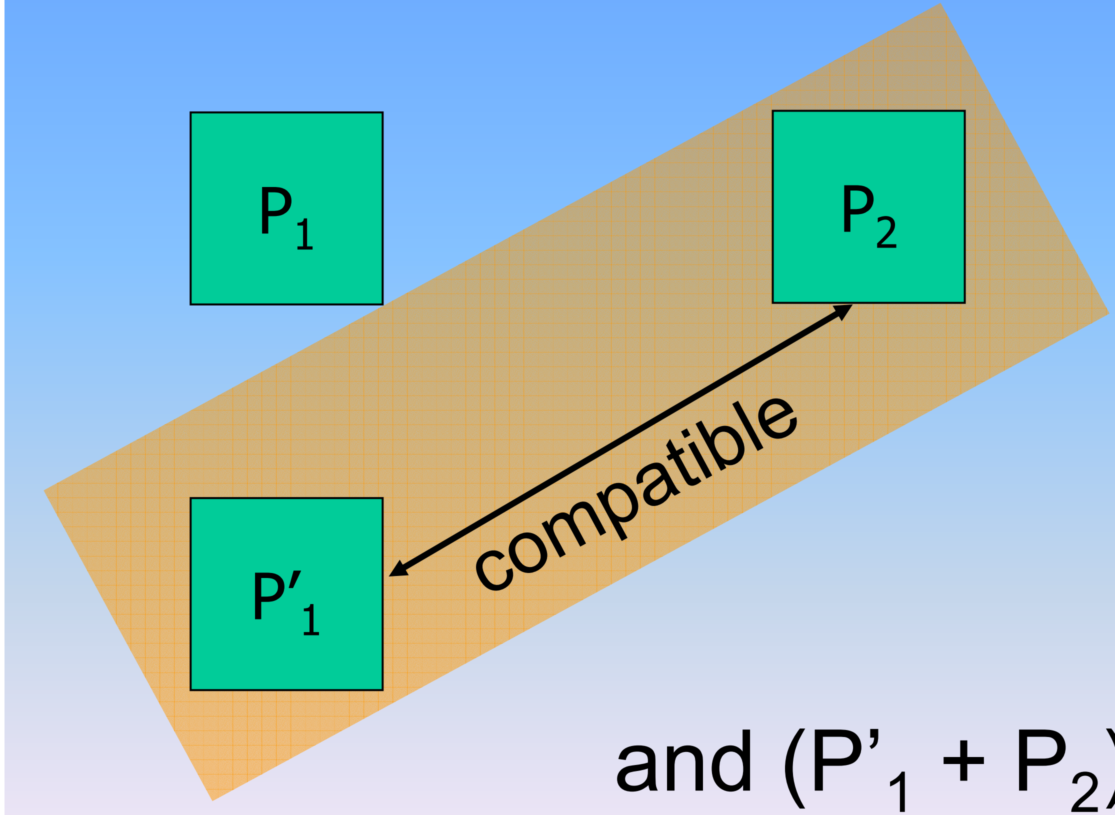
P_1 compatible P_2

$\curlyvee$

P'_1

Compositional Refinement

Then



and $(P'_1 + P_2) \preccurlyeq (P_1 + P_2)$

Refinement Preserves Specifications

If P $\models \phi$, then for all P' such that P' $\preccurlyeq$ P, we have P' $\models \phi$.

Existence of Environment

$P \models \phi$ if and only if there exists an environment E of P, such that $(P + E) \models \phi$.

If P_1 and P'_1 and P_2 are such that $P'_1 \preccurlyeq P_1$, and P_1 is compatible with P_2, then for any specification ϕ if $(P_1 + P_2) \models \phi$ then $(P'_1 + P_2) \models \phi$.

Summary

Three classes of interfaces for web services, allowing to reason about:

- Compatibility,
- Refinement, and
- Specification

for open systems.

Thanks!

Questions, Comments ?