Compatibility and replaceability analysis for timed web service protocols

Boualem Benatallah[1] Fabio Casati[2] Julien Ponge[3,1]
Farouk Toumani[3]

[1]Computer School of Engineering, The University of New South Wales, Sydney, Australia

[2]Hewlett-Packard Laboratories, Palo Alto, California, USA

[3]Laboratoire LIMOS, ISIMA, Clermont-Ferrand, France

21èmes journées Bases de Données Avancées
Saint-Malo

Agenda

Agenda

Agenda

Web services ?

"Les web services sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web." – Tutoriel IBM DeveloperWorks

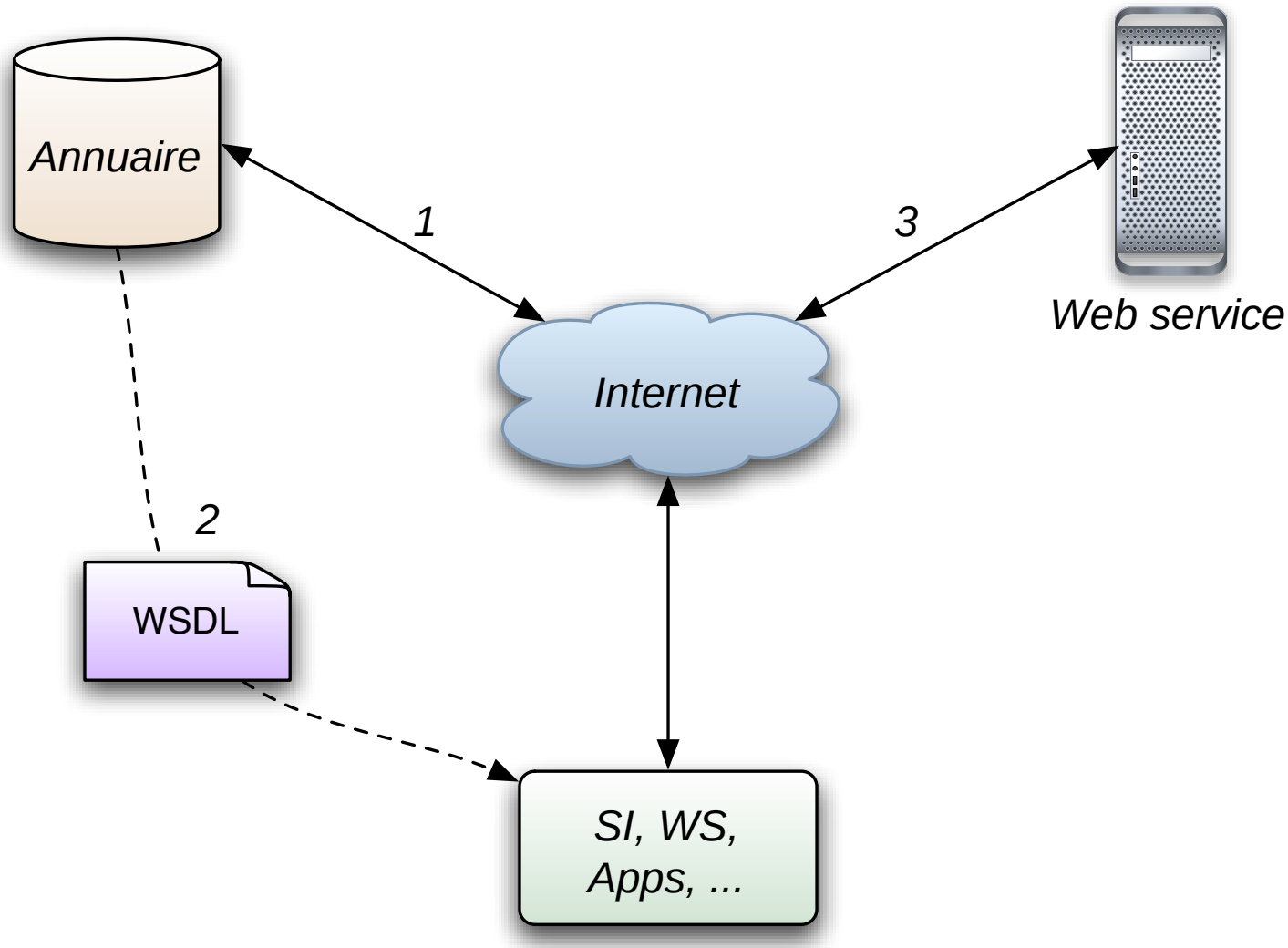
- Evolution middlewares classiques (MOM / RPC).
- Emplois de standards : XML, SOAP, HTTP(S), SMTP, ...
- *Services-Oriented Computing*, couplage faible.

Web services ?

"Les web services sont la nouvelle vague des applications web. Ce sont des applications modulaires, auto-contenues et auto-descriptives qui peuvent être publiées, localisées et invoquées depuis le web." – Tutoriel IBM DeveloperWorks

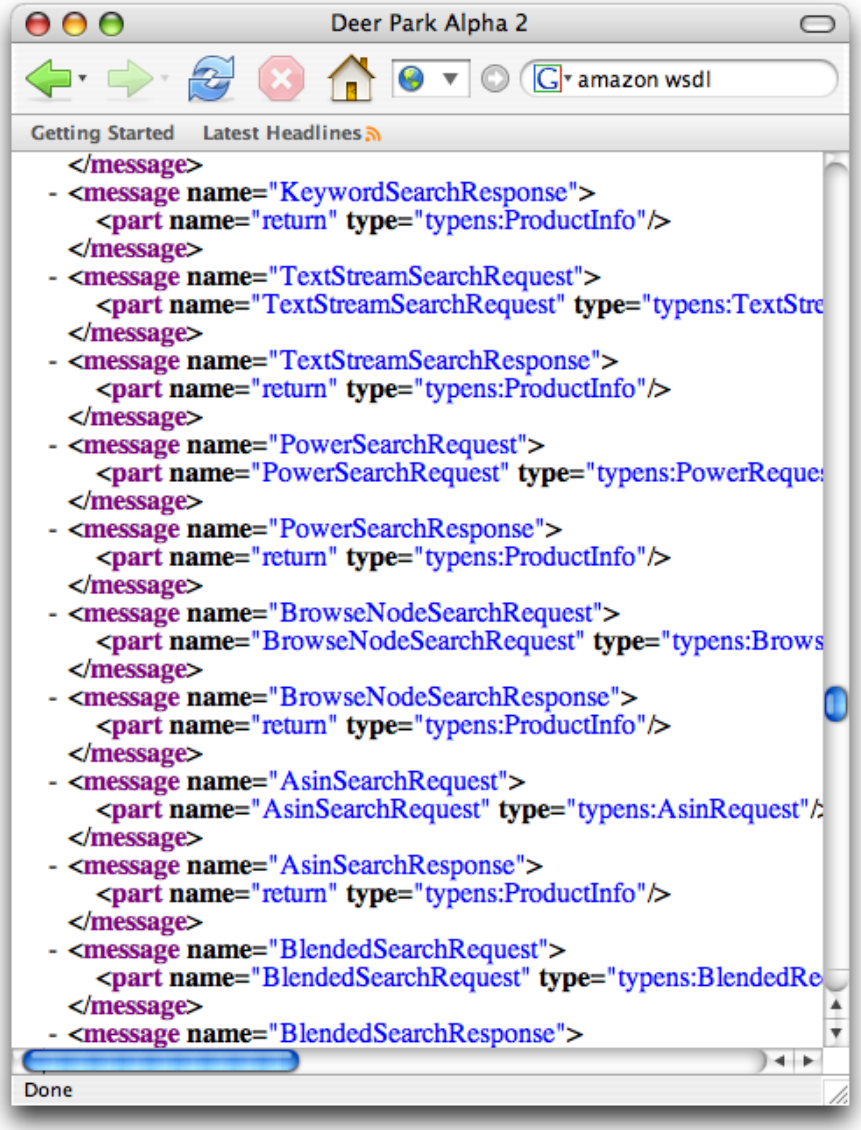
- Evolution middlewares classiques (MOM / RPC).
- Emplois de standards : XML, SOAP, HTTP(S), SMTP, ...
- *Services-Oriented Computing*, couplage faible.

Scénario usuel



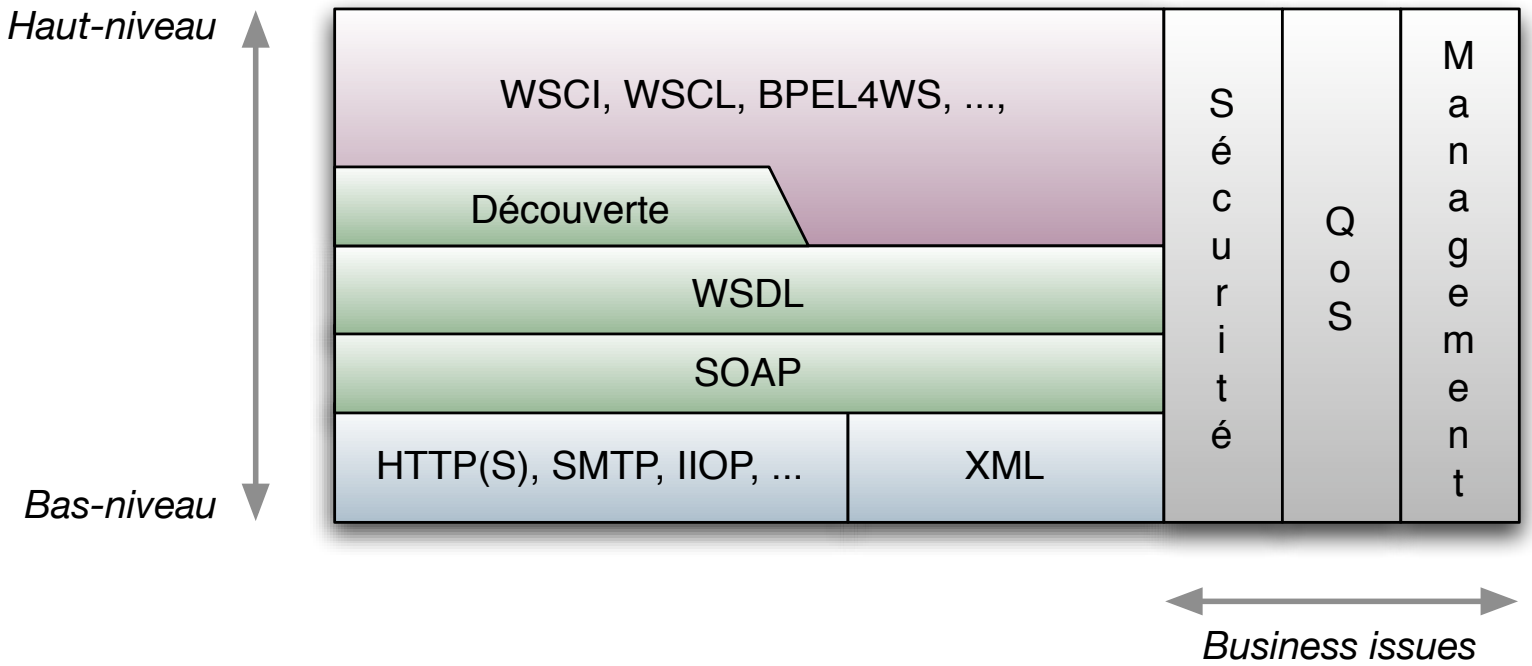
Pour le développeur ...

- SOAP / WSDL très bien acceptés, UDDI ... moins.
- Services riches (ex : Amazon) : beaucoup d'opérations.
- Difficultés pour "comprendre" le service.
- Standards bas-niveau, opérations manuelles.



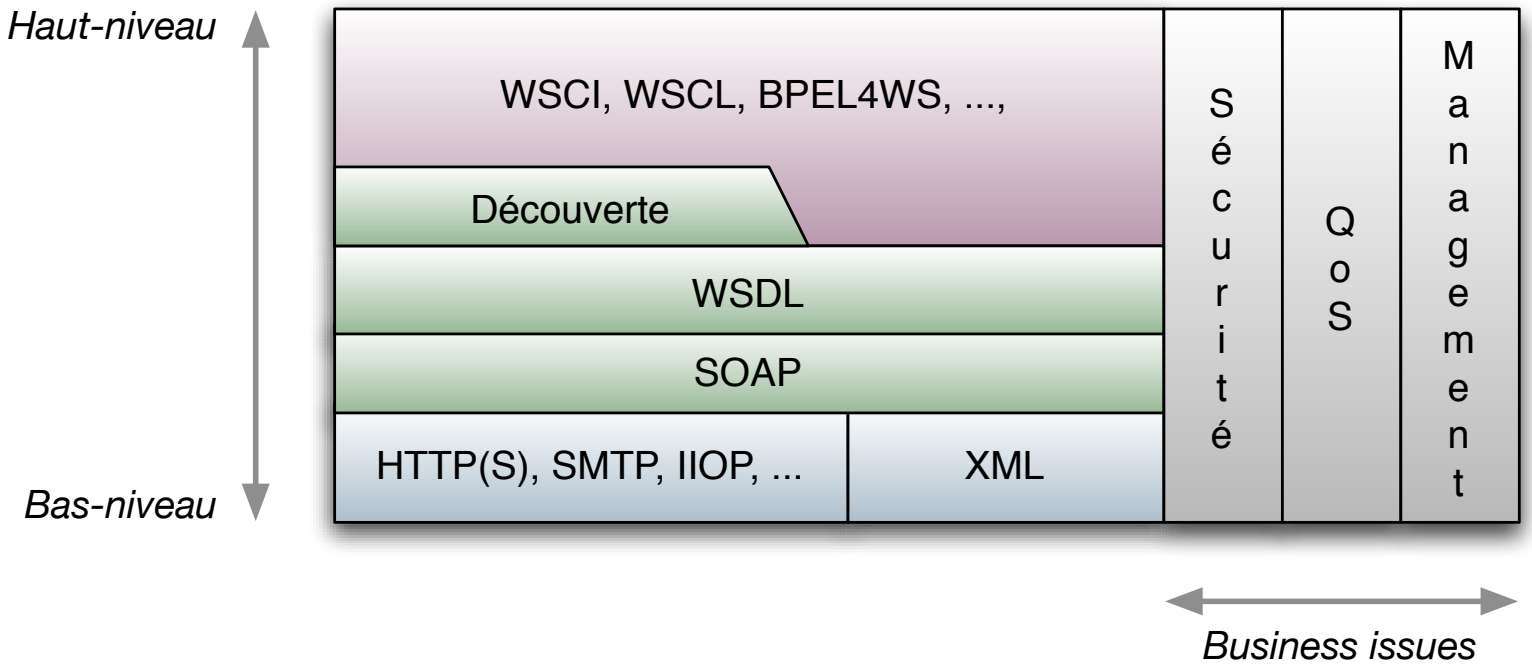
Agenda

(Une) Pile des web services



$\Longrightarrow$ Prolifération des standards (`WS-*`), orientés principalement implémentation.

(Une) Pile des web services

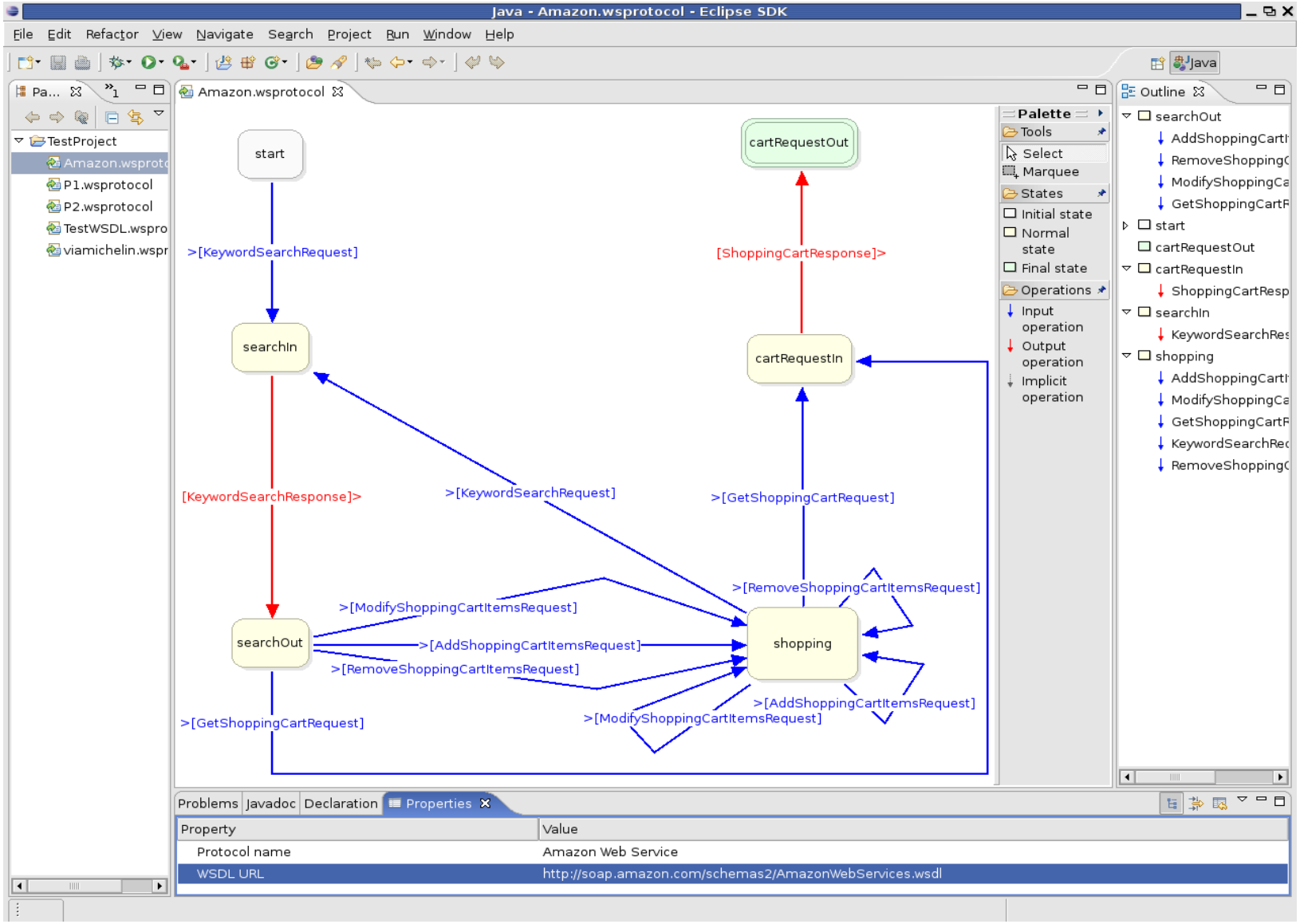


$\Longrightarrow$ Prolifération des standards (`WS-*`), orientés principalement implémentation.

ServiceMozaic

- Conception, développement et gestion de services.
- Ensemble d'outils orientés-modèles.
- J2EE + Eclipse.

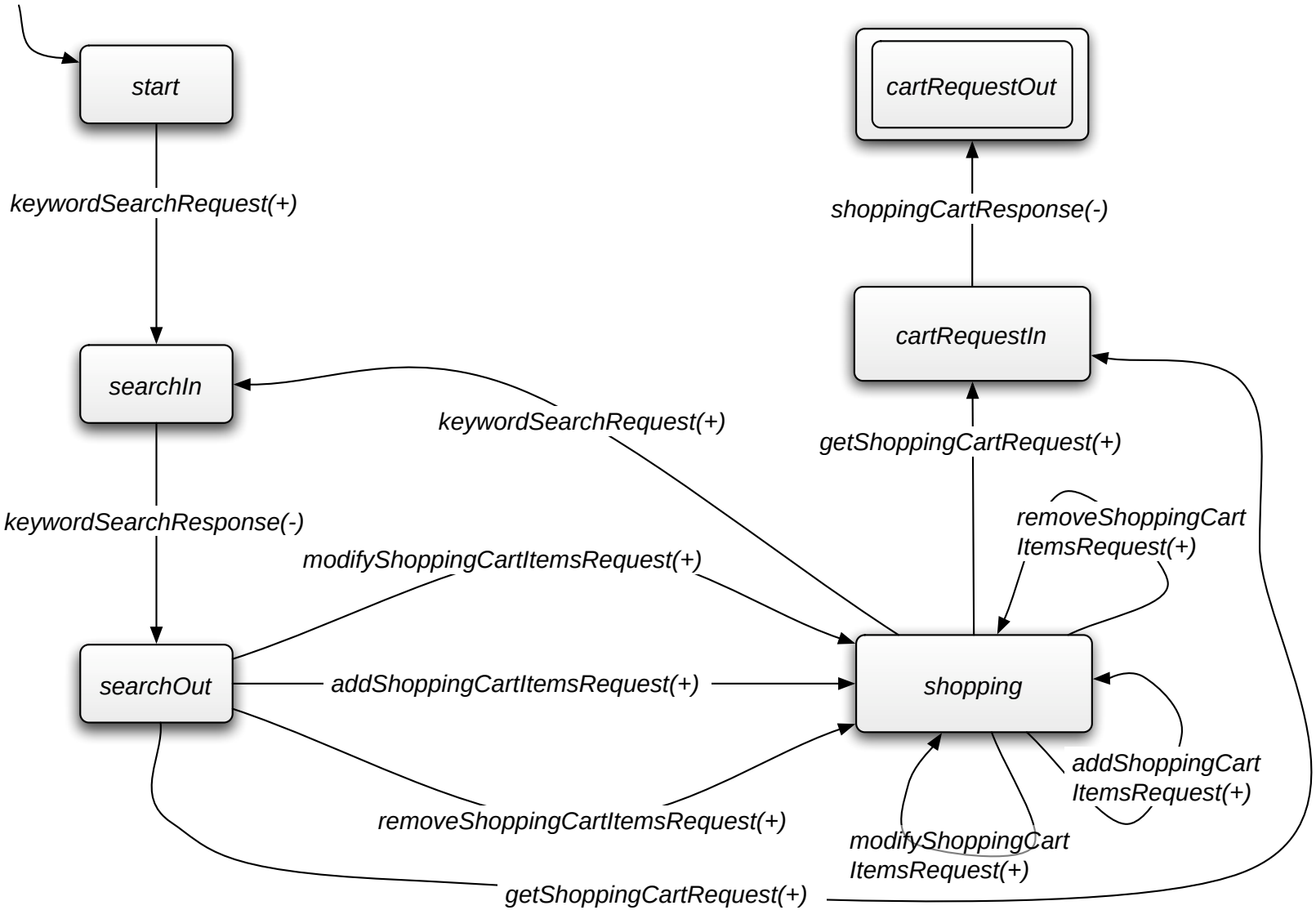
ServiceMozaic



Business protocols

- Représenter l'ensemble des conversations (*comportement externe*).
- Permettre des analyses via une algèbre.
- Automate déterministe :
  - états $\longleftrightarrow$ phases durant l'interaction client
  - transitions $\longleftrightarrow$ messages échangés (`in / out` $\sim +/-$).
- Instances multiples et concurrentes.

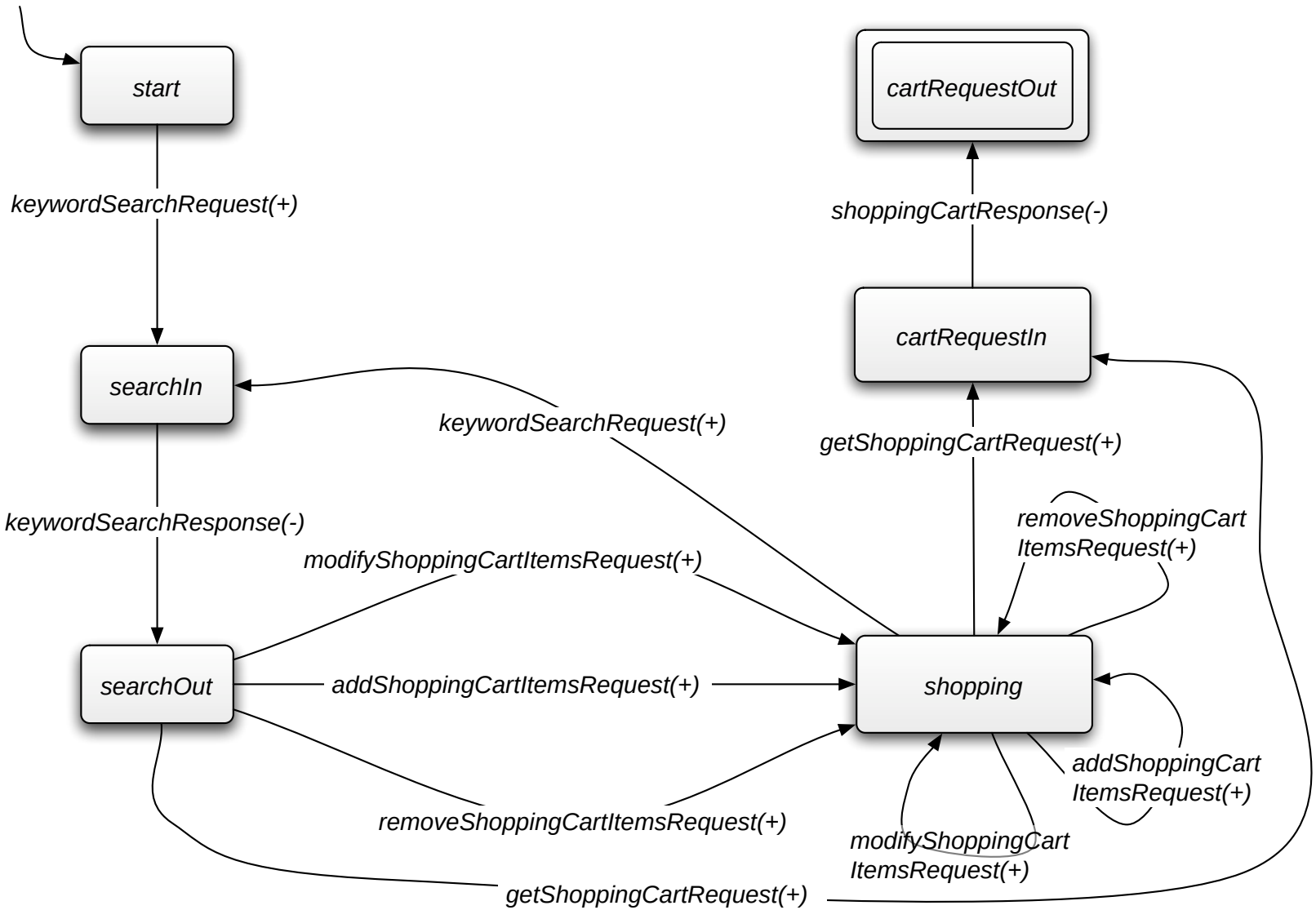
Exemple – Amazon AWS



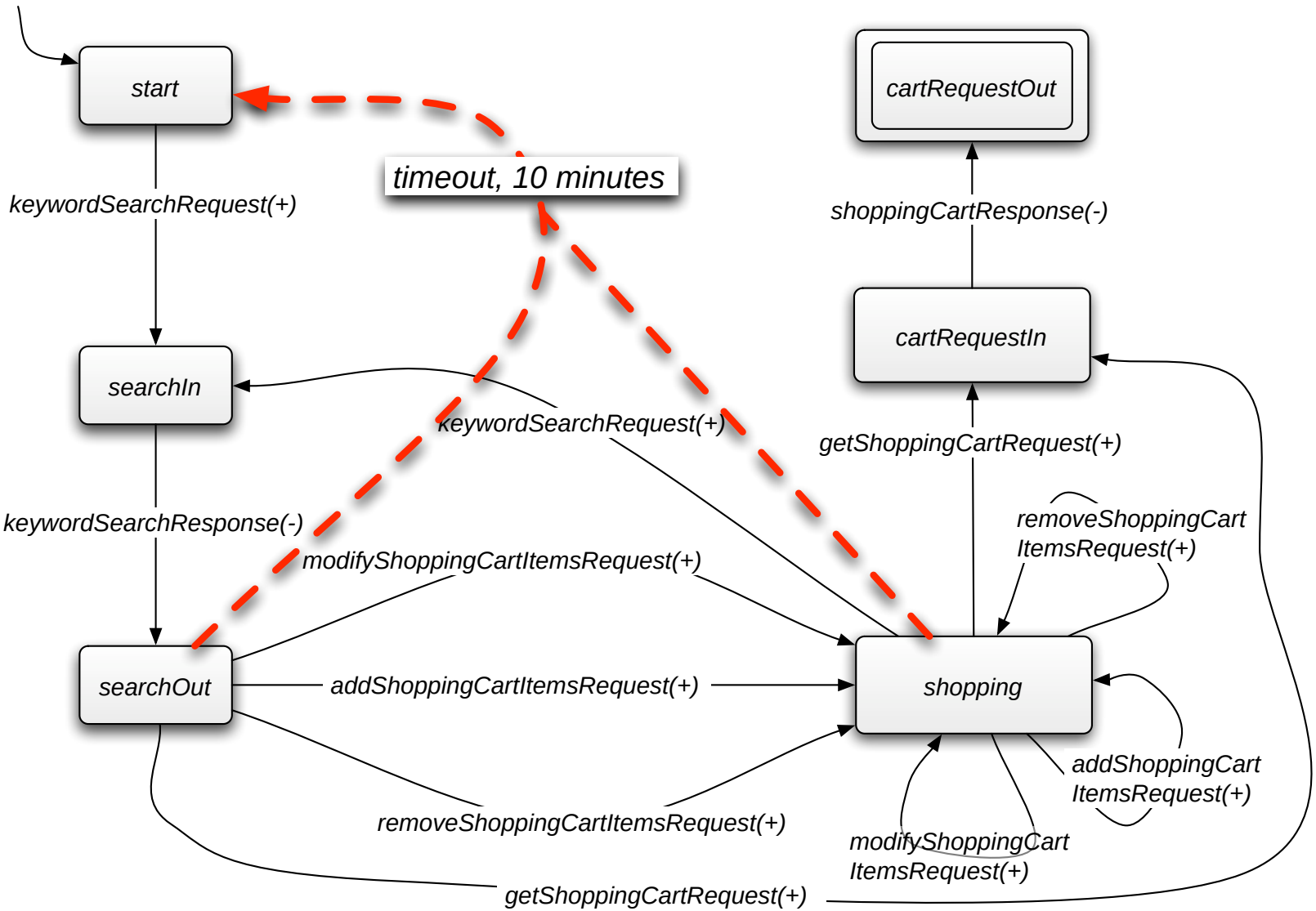
Identifier / modéliser des abstractions

- Compromis expressivité / complexité.
- Travaux précédents sur des portails *e-commerce*.
- Modèle de base simple et intuitif.
- Extensions pour les abstractions, ici le temps.

Intuition vis-à-vis du temps

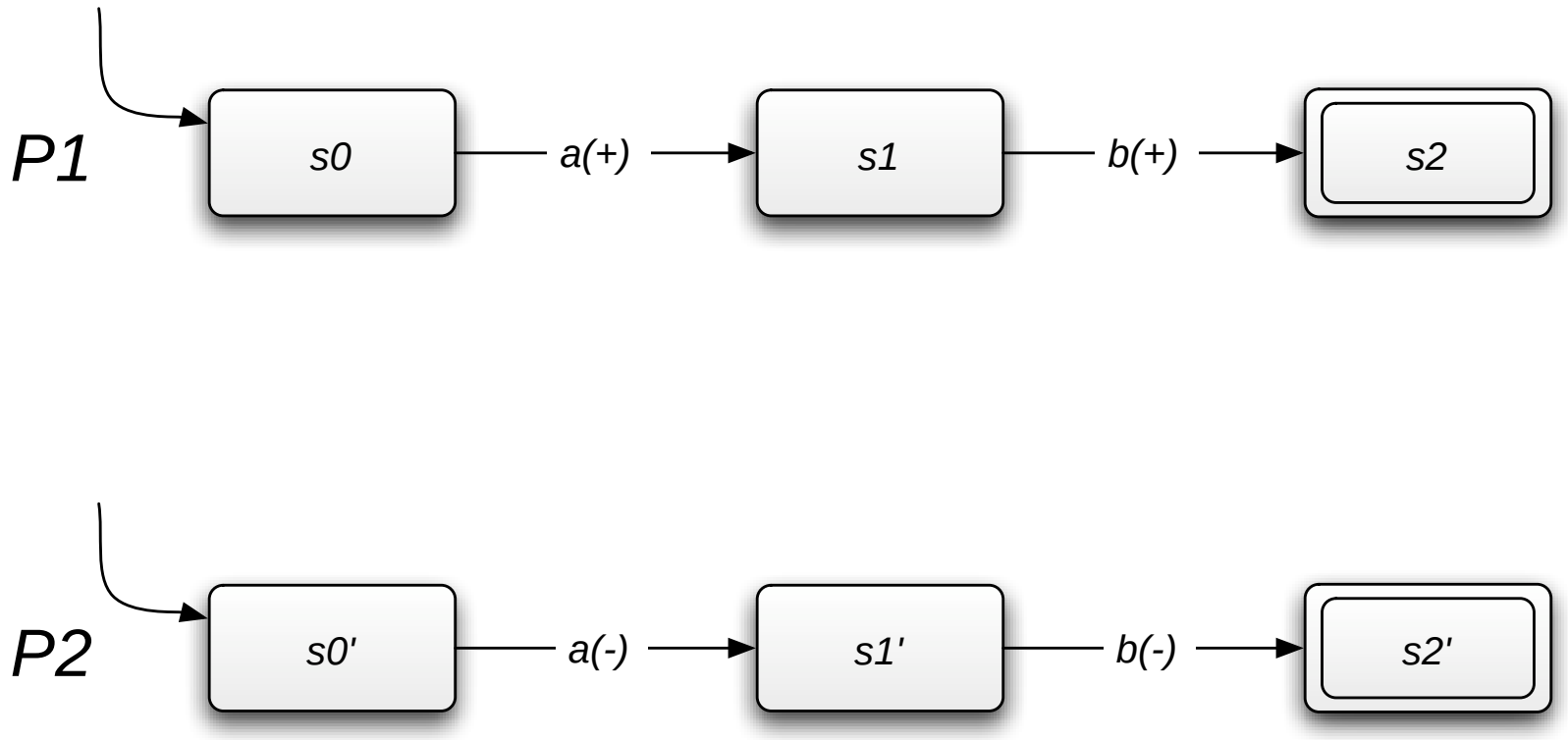


Intuition vis-à-vis du temps



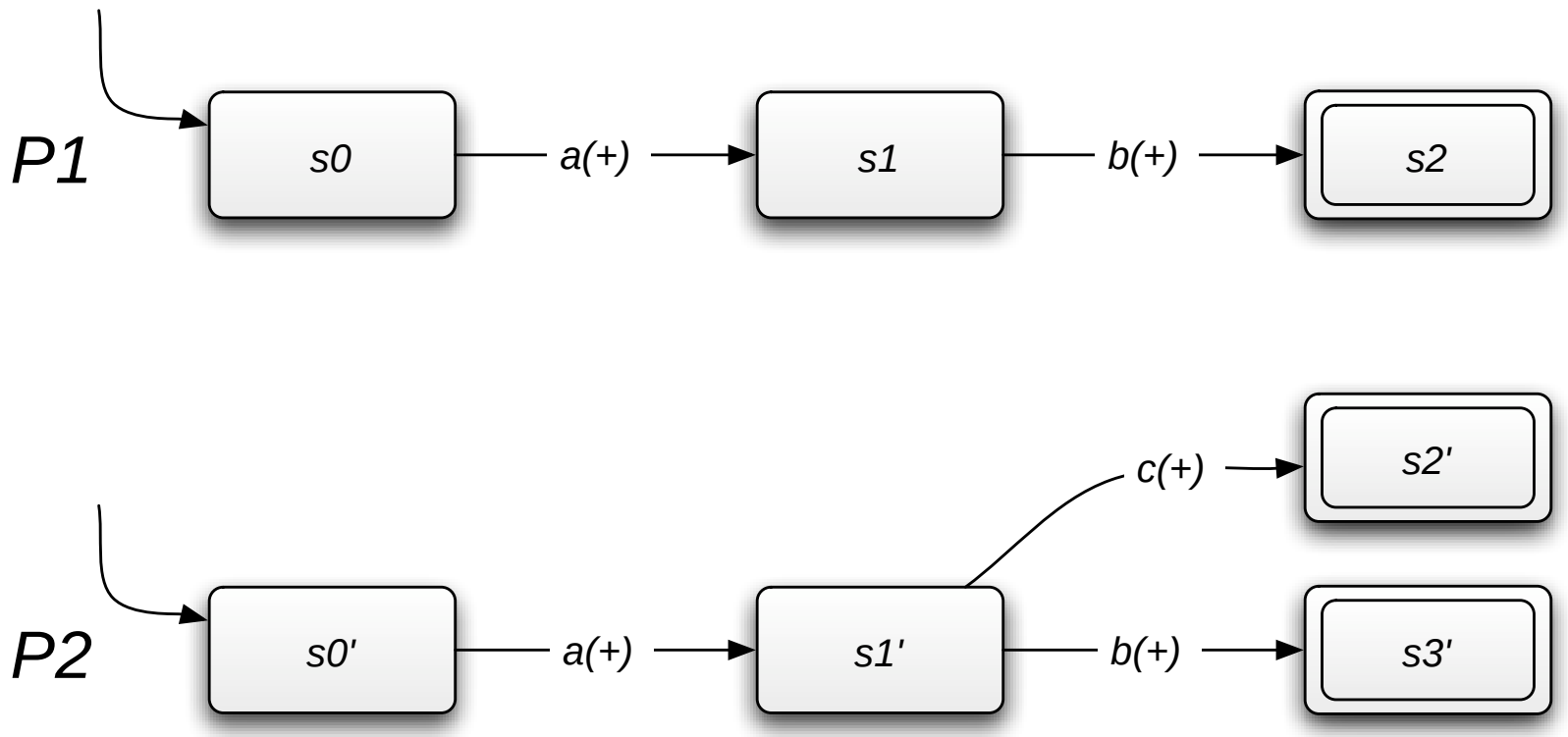
Agenda

Compatibilité



2 services peuvent converser

Remplaçabilité



1 service peut en remplacer un autre

Approche pour l'analyse

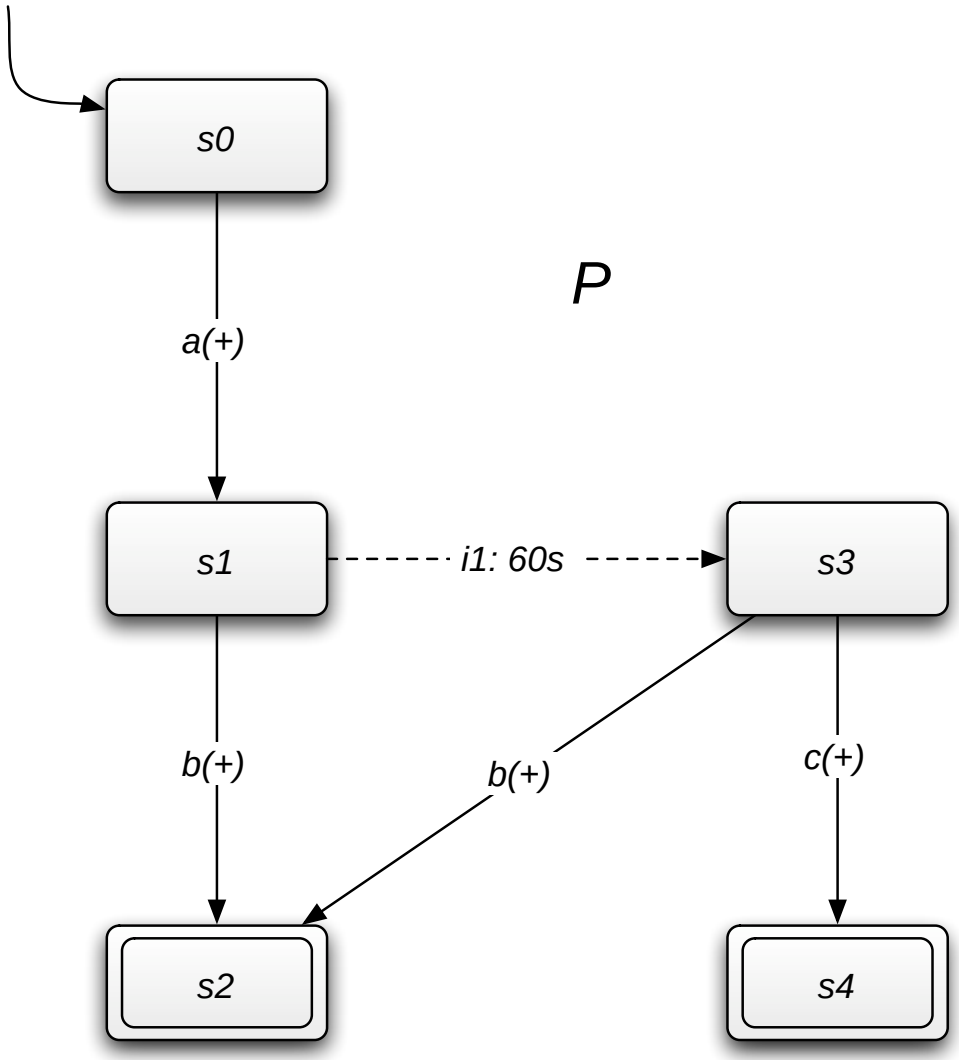
- Classes de compatibilité.
- Caractérisation à l'aide d'opérateurs de protocoles.
- Algorithmes *ad-hoc* pour les opérateurs.

Agenda

Agenda

Extension du modèle de base

- Introduction de transitions implicites.
- Modélisation intervalles de validité / expirations.



Formalisation

Web services business protocol

$\mathcal{P} = (\mathcal{S}, s_0, \mathcal{F}, \mathtt{M}, \mathcal{R})$

Timed web services business protocol :

- $\mathtt{M} = \mathtt{M}_e \cup \mathtt{M}_i$
- Pour $\mathcal{R}(s, s', m)$, $m \in \mathtt{M}_i$, définition de $Time(s, m) \rightarrow t \in \mathbb{Q}^{\geq 0}$.

Formalisation

Web services business protocol

$\mathcal{P} = (\mathcal{S}, s_0, \mathcal{F}, \mathtt{M}, \mathcal{R})$

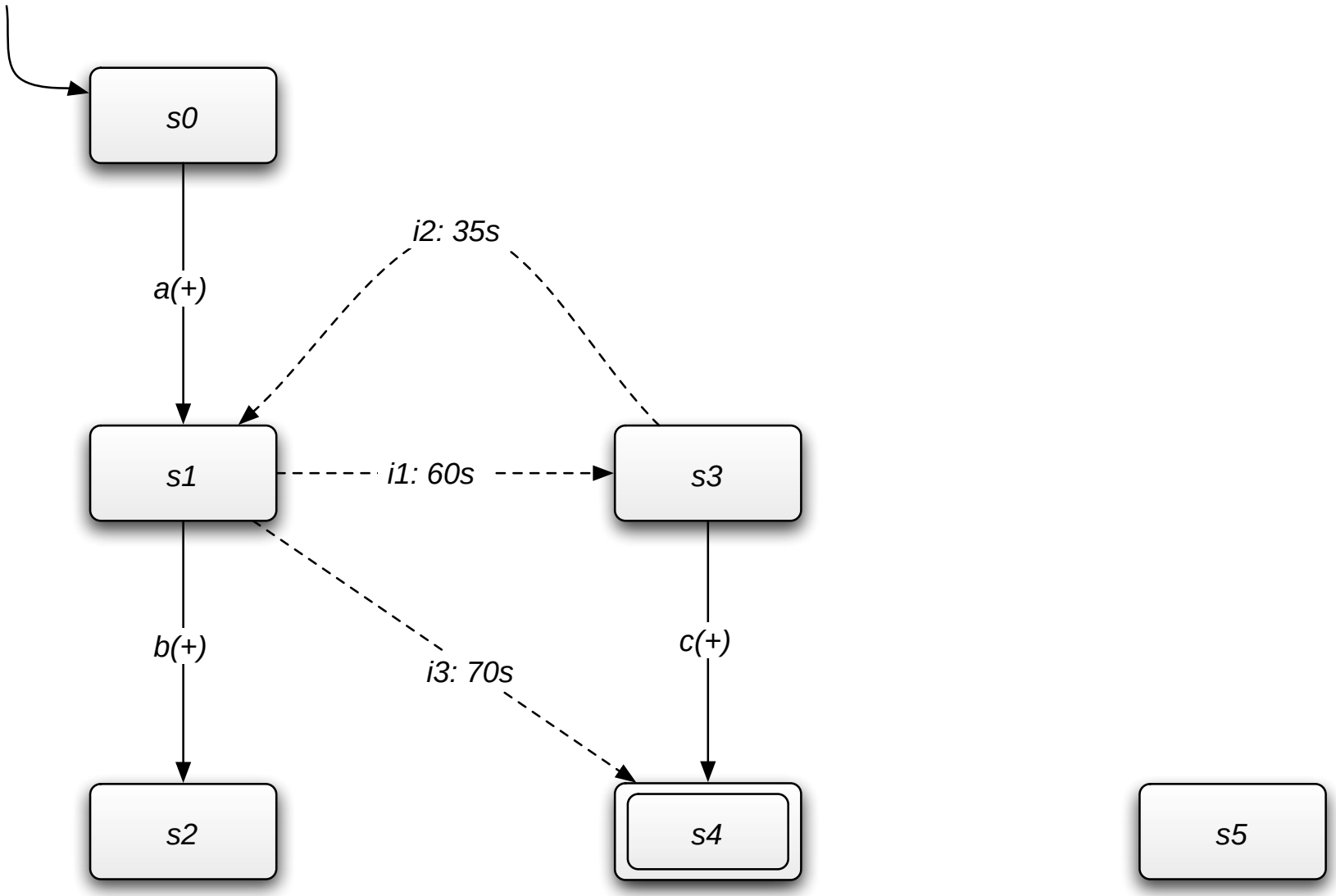
Timed web services business protocol :

- $\mathtt{M} = \mathtt{M}_e \cup \mathtt{M}_i$
- Pour $\mathcal{R}(s, s', m)$, $m \in \mathtt{M}_i$, définition de $Time(s, m) \rightarrow t \in \mathbb{Q}^{\geq 0}$.

Formalisation – cont.

- Protocoles déterministes.
- Au plus 1 transition implicite par état.
- Pas de *deadlocks*.
- Hypothèses :
  - transitions instantanées
  - temps relatif / entrée dans un état s
  - états atteignables
  - pas de circuits implicites.

Exemple : mauvais protocole



Sémantique

2 types de contraintes :

- conversations – *Linear time*

$$a(+) \cdot b(-) \cdot c(+)$$

- temporelles – traces temporisées

$$(a(+), 0) \cdot (b(-), 3) \cdot (c(+), 20)$$

On s'intéresse aux traces observables.

Sémantique

2 types de contraintes :

- conversations – *Linear time*

$$a(+) \cdot b(-) \cdot c(+)$$

- temporelles – traces temporisées

$$(a(+), 0) \cdot (b(-), 3) \cdot (c(+), 20)$$

On s'intéresse aux traces observables.

Agenda

Classes de compatibilité et remplaçabilité

- Compatibilité : totale et partielle.
- Remplaçabilité :
  - équivalence, partielle et subsomption
  - par rapport à un protocole client
  - par rapport à un rôle d'interaction

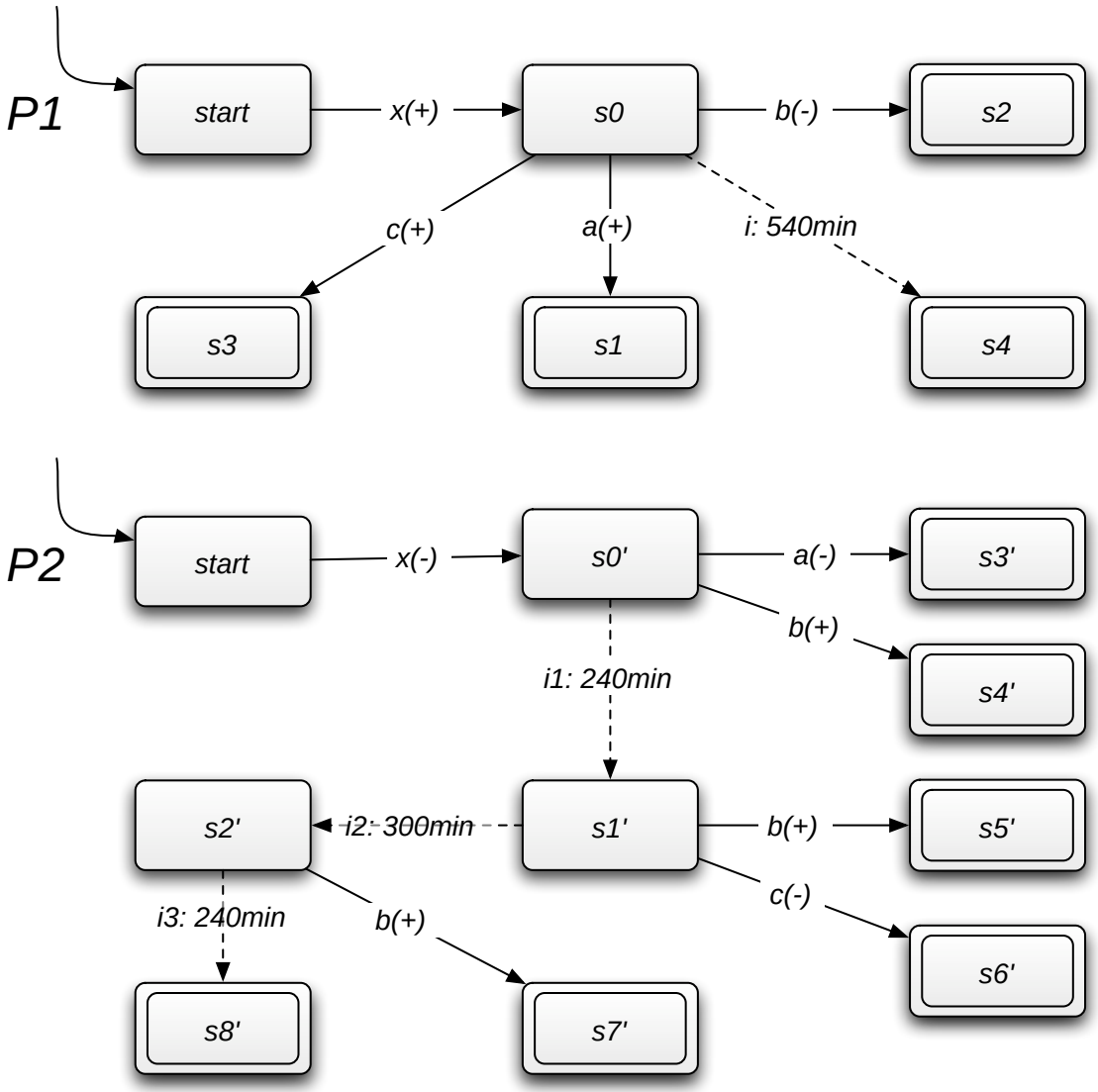
$\longrightarrow$ originalité : flexibilité.

Classes de compatibilité et remplaçabilité

- Compatibilité : totale et partielle.
- Remplaçabilité :
  - équivalence, partielle et subsomption
  - par rapport à un protocole client
  - par rapport à un rôle d'interaction

$\longrightarrow$ originalité : flexibilité.

Difficulté



$\rightarrow \mathcal{P}_1$ et $\mathcal{P}_2$ sont compatibles !

$\rightarrow$ Transitions implicitement disponibles.

s'_0, s_0 :

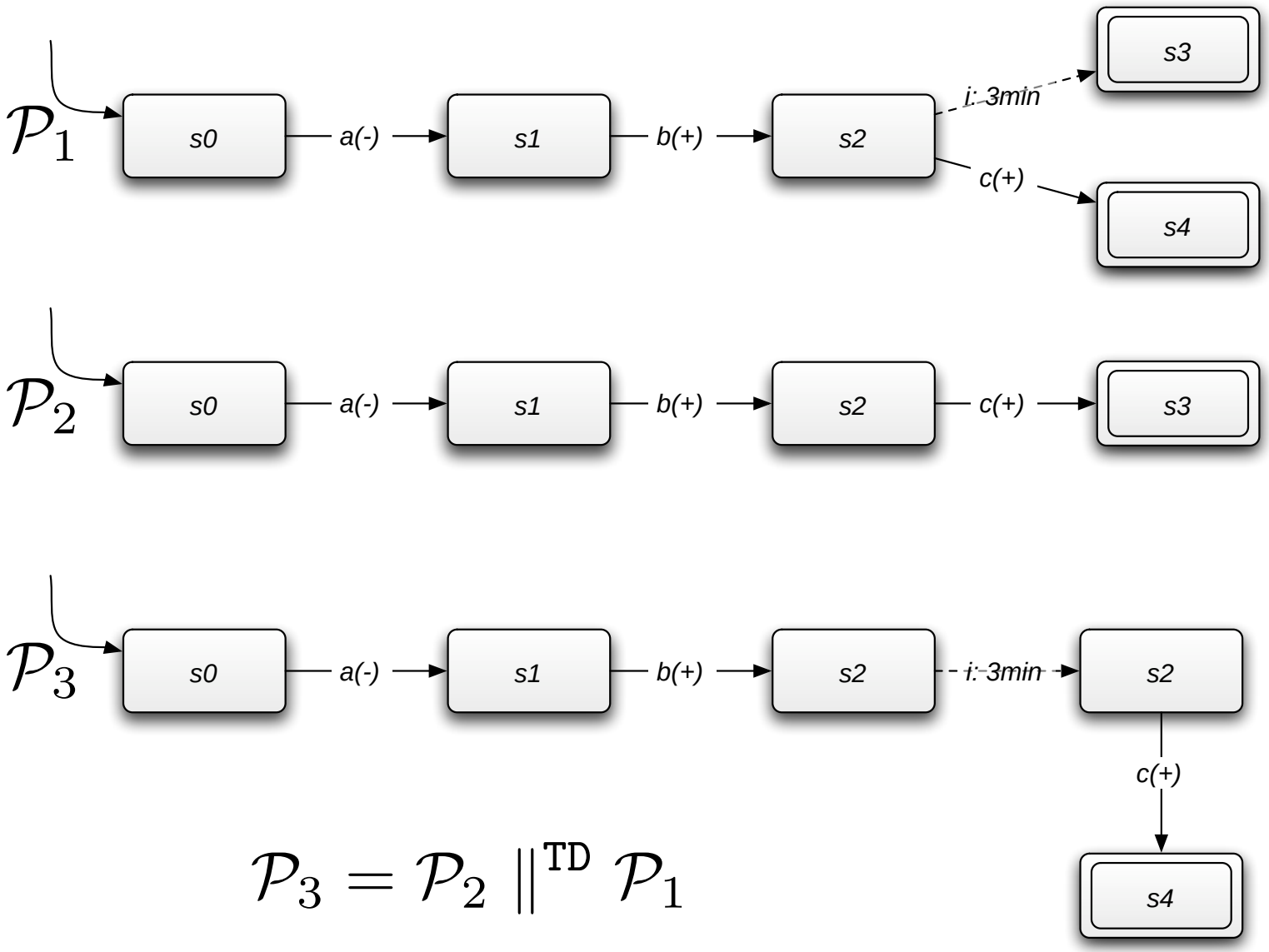
$$(P_1) \begin{cases} a : [0min, 540min] \\ b : [0min, 540min] \\ c : [0min, 540min] \end{cases}$$

$$(P_2) \begin{cases} a : [0min, 240min] \\ b : [0min, 780min] \\ c : [240min, 540min] \end{cases}$$

Opérateurs

- Timed compatible composition : $\|^{\mathtt{TC}}$
- Timed intersection : $\|^{\mathtt{TI}}$
- Timed difference : $\|^{\mathtt{TD}}$
- Projection : $[\mathcal{P}_1 \|^{\mathtt{TC}} \mathcal{P}_2]_{\mathcal{P}_1}$

Exemple : différence temporisée



$$\mathcal{P}_3 = \mathcal{P}_2 \parallel^{\mathtt{TD}} \mathcal{P}_1$$

Caractérisation

- Caractérisation des classes de compatibilité et remplaçabilité par les opérateurs.
- Ex : $TRepl_{\mathcal{P}_C}(\mathcal{P}_1, \mathcal{P}_2)$

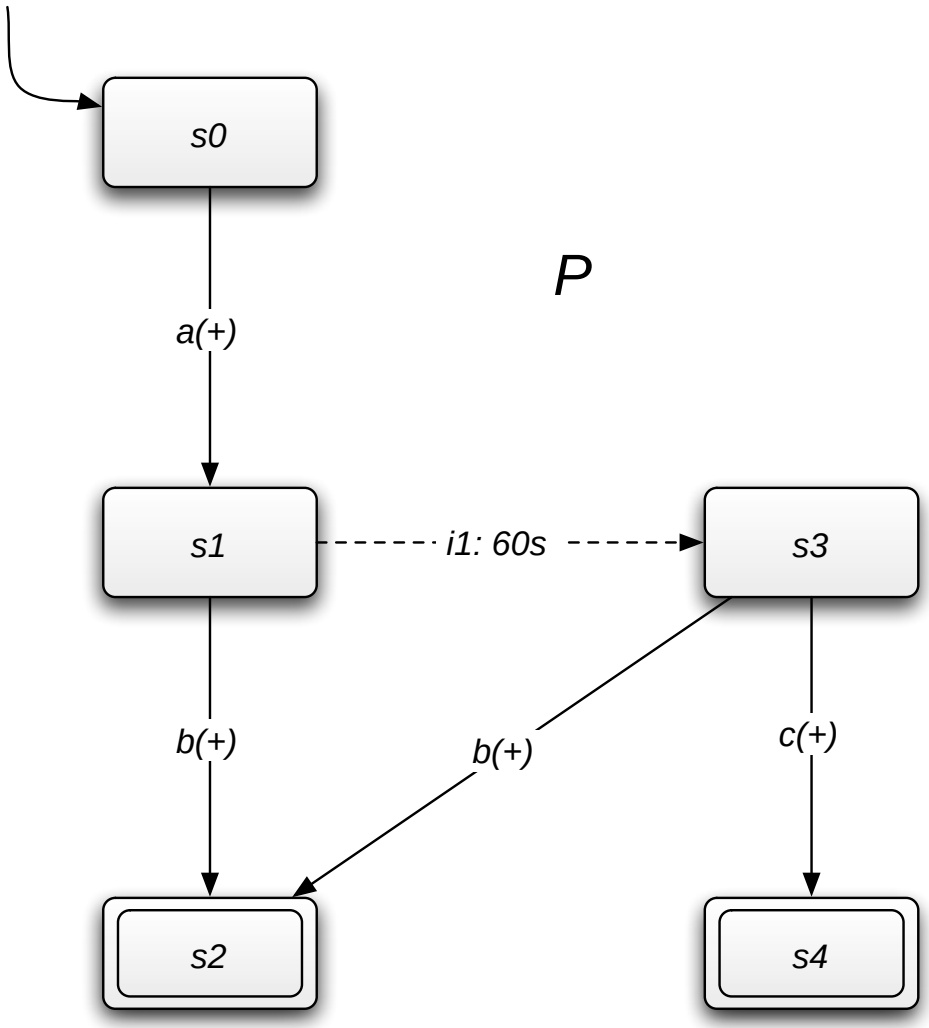
$$\mathcal{P}_C \parallel^{\mathtt{TC}} (\mathcal{P}_2 \parallel^{\mathtt{TD}} \mathcal{P}_1) = \emptyset$$

Agenda

Résultats

- Algorithmes polynomiaux pour les opérateurs.
- Algorithme de fermeture de transitions implicite.
- *Time-state space* : disponibilités temporelles d'une transition depuis un état.

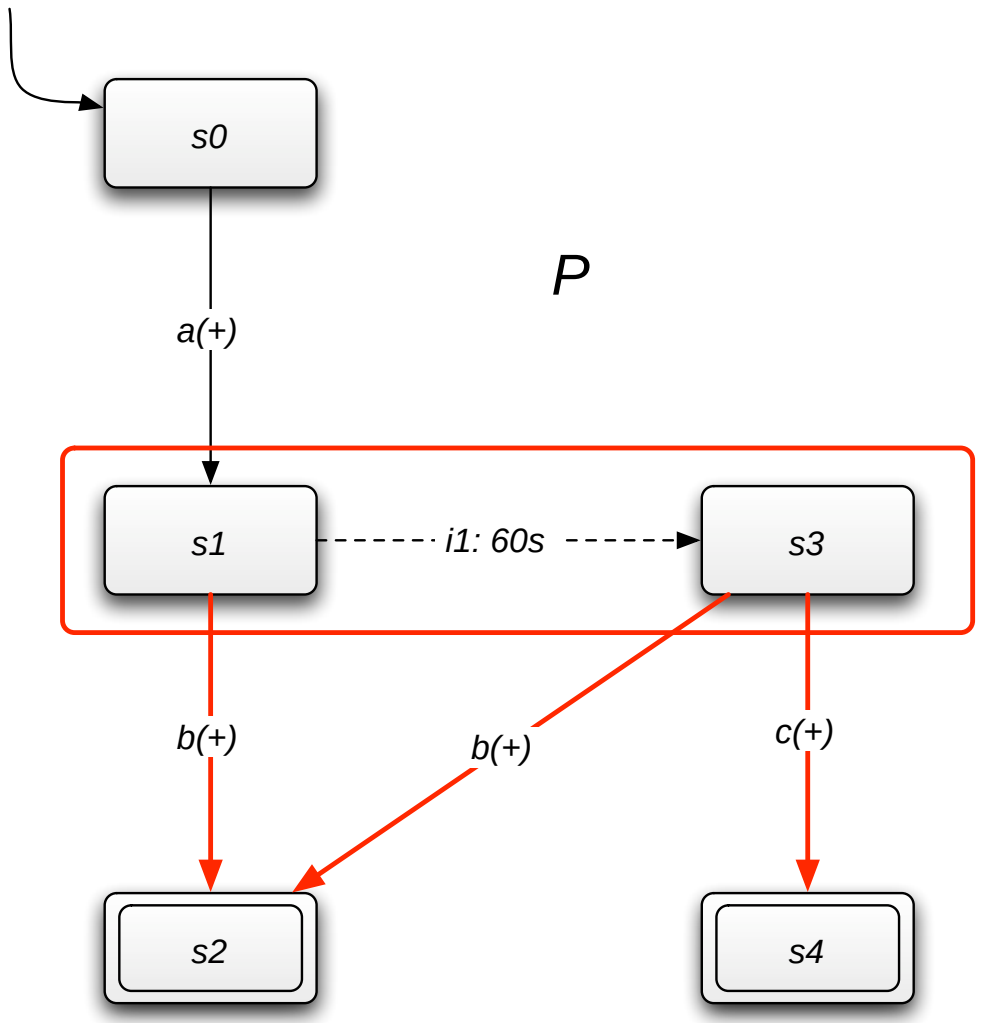
Expliciter par fermeture



Depuis s_1 :

b : $([0, +\infty[, s_1)$
c : $([60, +\infty[, s_1)$

Expliciter par fermeture



Depuis s_1 :

b : $([0, +\infty[, s_1)$
c : $([60, +\infty[, s_1)$

Agenda

Contribution

- Modèle temporisé pour le comportement externe de services.
- Analyse flexible de compatibilité et remplaçabilité.
- Opérateurs génériques + algorithmes.

Perspectives

- Limite : contraintes par rapport au dernier message.
- Lien avec les *timed automata*.
- Intégration dans *ServiceMozaic*.

Merci pour votre attention.