Sémantique formelle et synthèse de client pour services Web

Séminaire « *Services Web* »
24 Janvier 2006

Sylvain Rampacek
sylvain.rampacek@univ-reims.fr
CReSTIC – LAMSADE

Plan

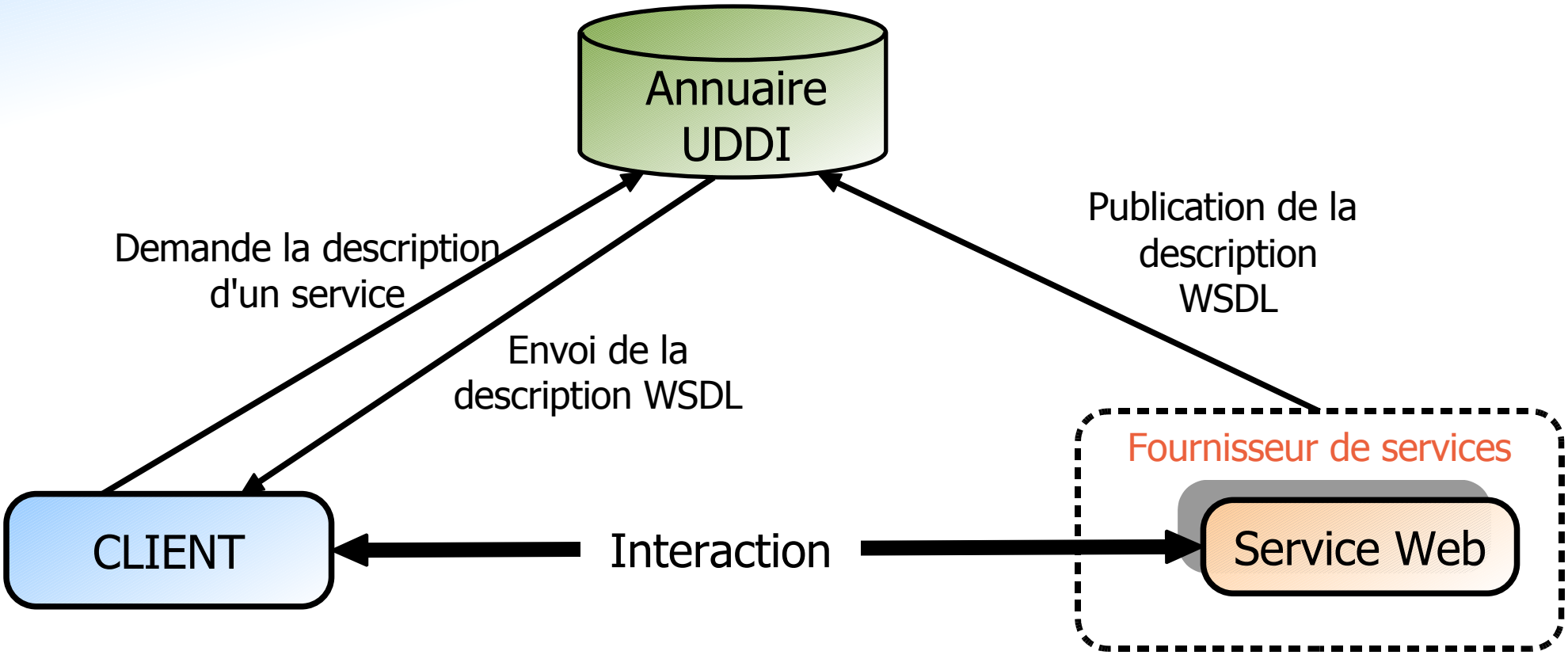
- Introduction
  - Services Web
  - Description de la plate-forme
- Sémantique formelle
  - Une sémantique formelle pour les services Web
  - Les automates temporisés
- Relation d'interaction
  - Ambiguïté
  - Génération d'un client
- Conclusion

Introduction

Contexte des Services Web

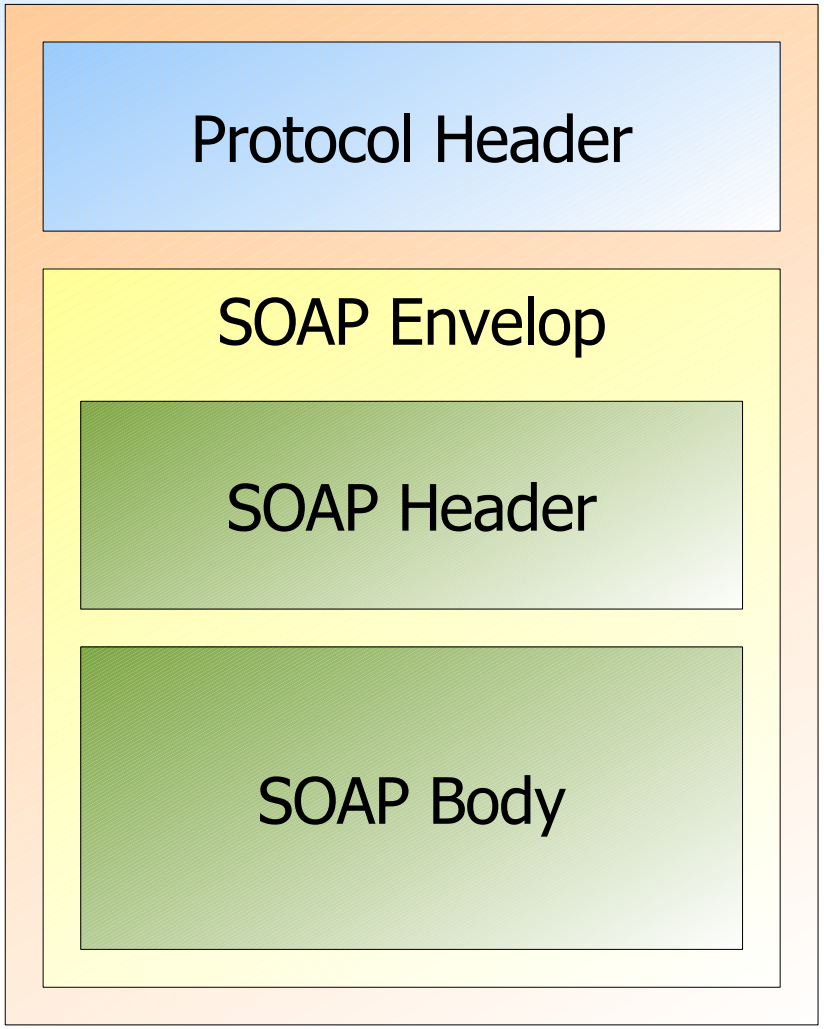
- Systèmes distribués (un service ≠ un serveur)
- Interopérabilité (XML, SOAP, WSDL, ...)
- Gestion de l'hétérogénéité :
  - Niveau/Couche supplémentaire (conserve la couche métier)
  - Évolution des systèmes distribués à objets
  - *Service Oriented Architecture* (SOA)

Architecture des Services Web



SOAP

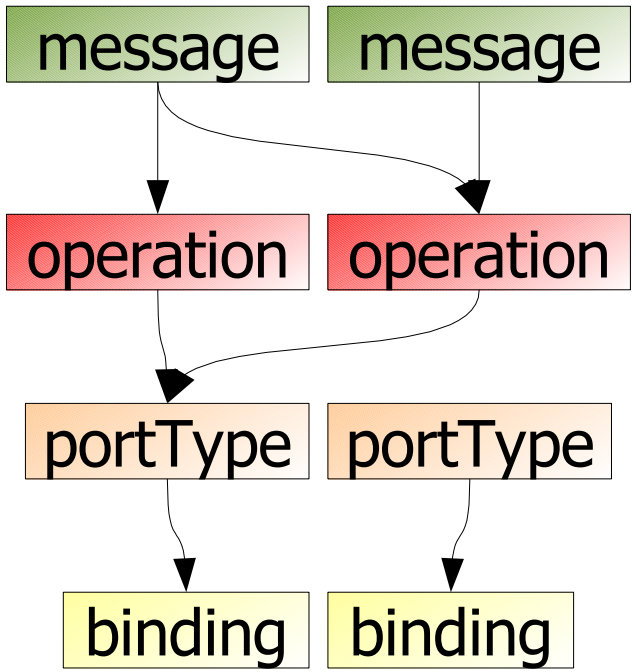
- *Simple Object Access Protocol* -



- basé sur XML
- représente les données lors des échanges
- *protocol header* : pour le niveau transport
- SOAP Envelop :
  - SOAP Header : information à propos des noeuds intermédiaires
  - SOAP Body : « données » dans un langage spécifique

WSDL

- *Web Services Description Language* -

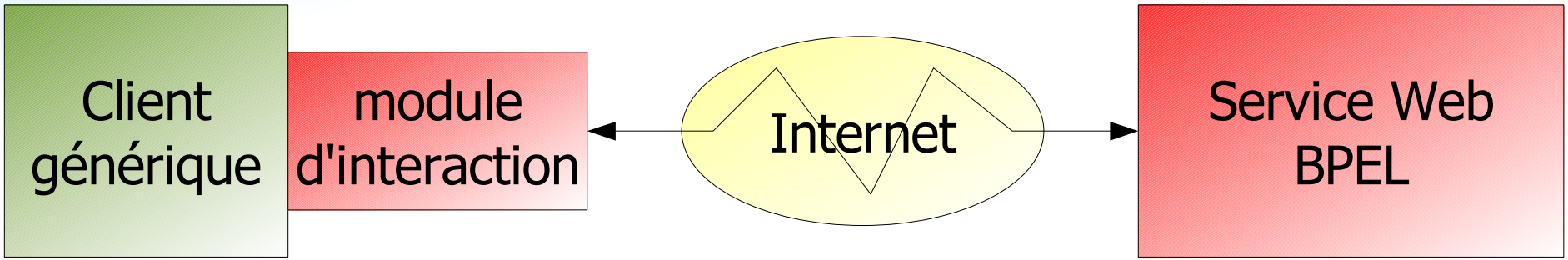


- Interface du service (sens C++/JAVA)
- basé sur XML
- Décrit :
  - les espaces de noms
  - les messages
  - les opérations (composition de messages en entrée et en sortie)
  - portType (port de communication)
  - liaison (lien entre les opérations WSDL et les opérations SOAP)

BPEL

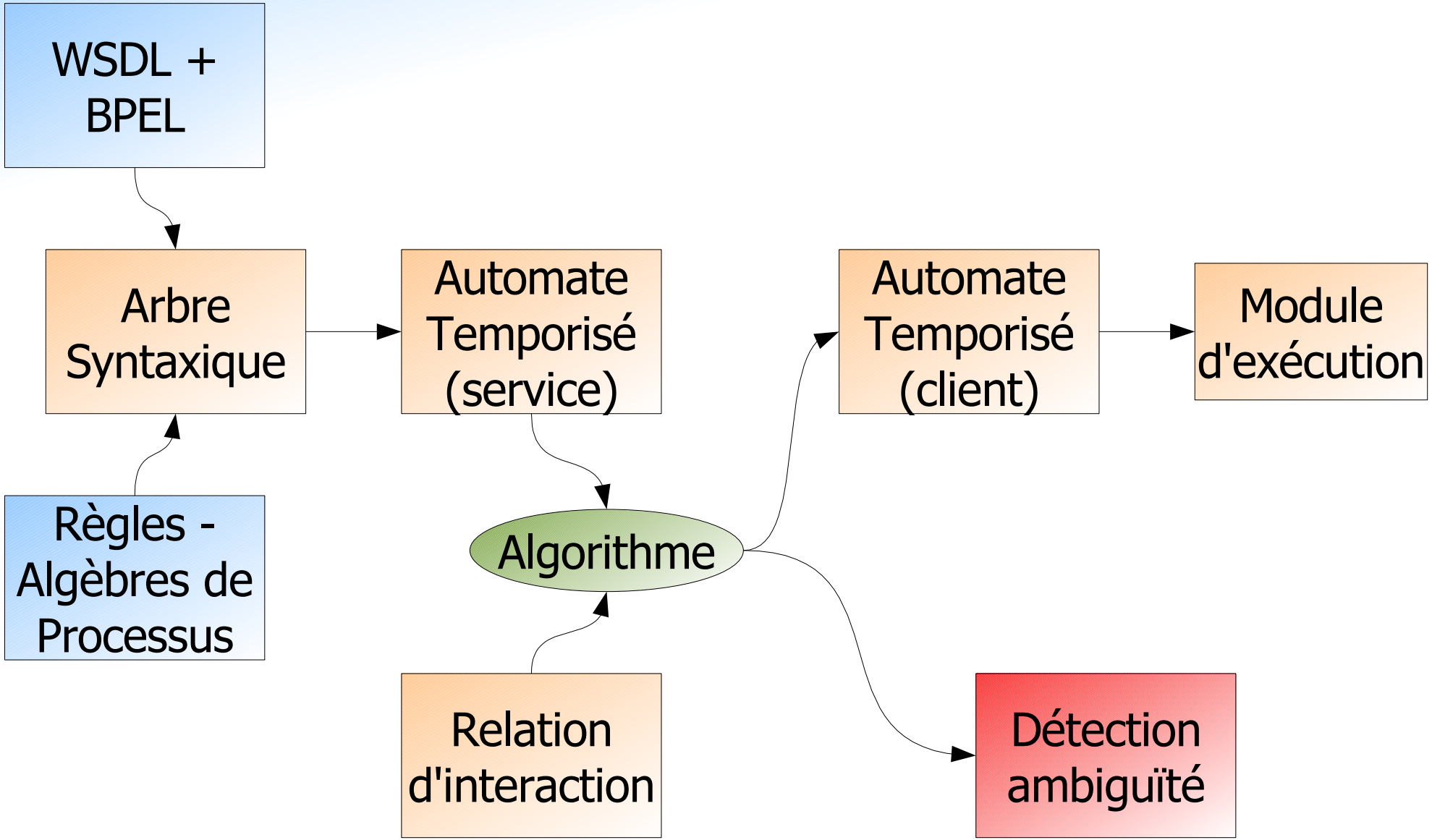
- Langage de description comportementale
  - en complément de la description WSDL
- Historique :
  - Version 1.0 (Juillet 2002) – BPEL4WS
    - IBM, BEA, Microsoft
    - fusion/évolution de XLANG, BPML, ...
    - actions élémentaires (*while*, *switch*, *scope*, *flow*, etc.)
    - mécanisme de compensation (transactions longues)
  - Version 1.1 (Mai 2003) – BPEL4WS
  - Version 2.0 (2006 ?) – WS-BPEL ou BPEL
    - standard Oasis

Contexte du client générique



- Développement d'un client générique
  - Invocation de services Web BPEL
- Problème
  - Interaction possible ?

Étapes de la génération



Une sémantique formelle pour les processus BPEL abstraits

Introduction

Les Automates Temporisés

Les règles sémantiques

Sémantique formelle

- Nécessaire pour la construction d'un client approprié et son exécution
  - Relation d'interaction
  - Génération contrôlée du client
  - Composition formelle (par la suite)
- Deux sémantiques temporelles
  - Un service vu comme une algèbre de processus temporisés
  - Sémantiques associées :
    - TIOTS (temps discret)
    - Automates Temporisés (temps dense)

BPEL et sémantique formelle

- BPEL fournit un ensemble de constructeurs décrivant de façon modulaire le comportement observable d'un processus abstrait
- Le temps est présent dans de nombreux constructeurs BPEL.
- Processus similaires à ceux des algèbres de processus
  - temps dense ?
- Notre approche
  - association d'un processus abstrait à un automate temporisé

Une sémantique formelle pour les processus BPEL abstraits

Introduction

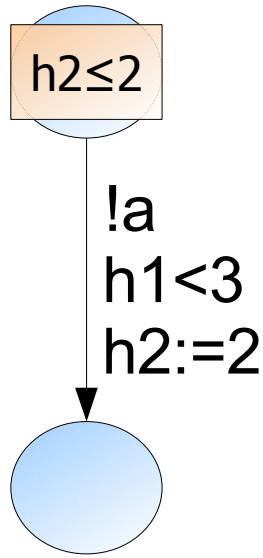
Les Automates Temporisés

Les règles sémantiques

Les automates temporisés

Définition

- Automate Temporisé (AT) de Alur et Dill (1994)
  - « automate avec des horloges »
- Les horloges mesurent le temps écoulé depuis leur dernière ré-initialisation
  - une horloge est active après son initialisation
- Un arc liant deux états comporte :
  - une action *a*
  - une garde *g*
  - un ensemble d'horloges à réinitialiser *r*



L'alphabet pour BPEL

- envoi/réception de messages $!o[m]$ / $?o[m]$
- expiration de *timeout* to
- action interne (et immédiate) τ
- action de terminaison $\surd$

L'automate temporisé d'un processus BPEL *(les états)*

- Chaque état de l'AT est associé avec un processus abstrait BPEL
- Les arcs symbolisent alors les différentes actions qu'un processus peut réaliser.
  - en exécutant une action, un processus devient un nouveau processus, associé à un nouvel état
- L'ensemble des états est obtenu par transformation successive du processus initial
- L'état initial correspond au processus métier dans sa globalité

Une sémantique formelle pour les processus BPEL abstraits

Introduction
Les Automates Temporisés
Les règles sémantiques

Format d'une règle de sémantique formelle

$$[op_x] := \frac{B_{\exp}^{\alpha_i}(\{P_{o(i)} \rightarrow P'_{o(i)}\})}{P \xrightarrow{L_{\exp}(\{\alpha_i\})} N_{\exp}(P, \{P'_{o(i)}\})} \; avec\ garde(\{\alpha_i\})$$

Sémantique formelle

processus élémentaires

processus *empty*

$$empty \xrightarrow{\surd} 0$$

processus *operation*

$$*o[m] \xrightarrow{*m} empty$$

processus *throw*

$$throw[e] \xrightarrow{e} 0$$

Sémantique formelle

constructeurs « séquentiels »

processus *sequence*

$$\forall\, a \neq \surd \;\; \frac{P \xrightarrow{a} P'}{P ; Q \xrightarrow{a} P' ; Q}$$

$$\forall\, a \;\; \frac{P \xrightarrow{\surd} \;\wedge\; Q \xrightarrow{a} Q'}{P ; Q \xrightarrow{a} Q'}$$

processus *switch*

$$switch[\{P_i\}] \xrightarrow{\tau} P_i$$

processus *while*

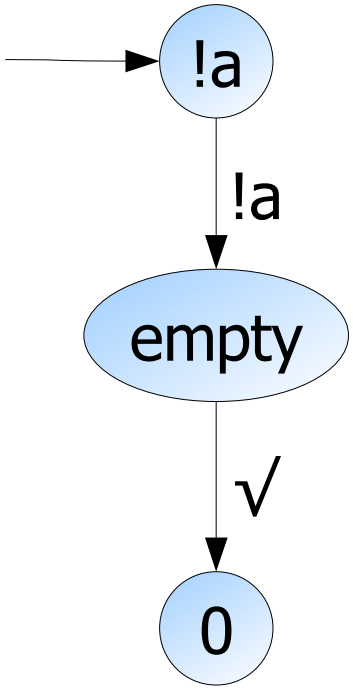
$$while[P] \xrightarrow{\tau} empty$$

$$while[P] \xrightarrow{\tau} P ; while[P]$$

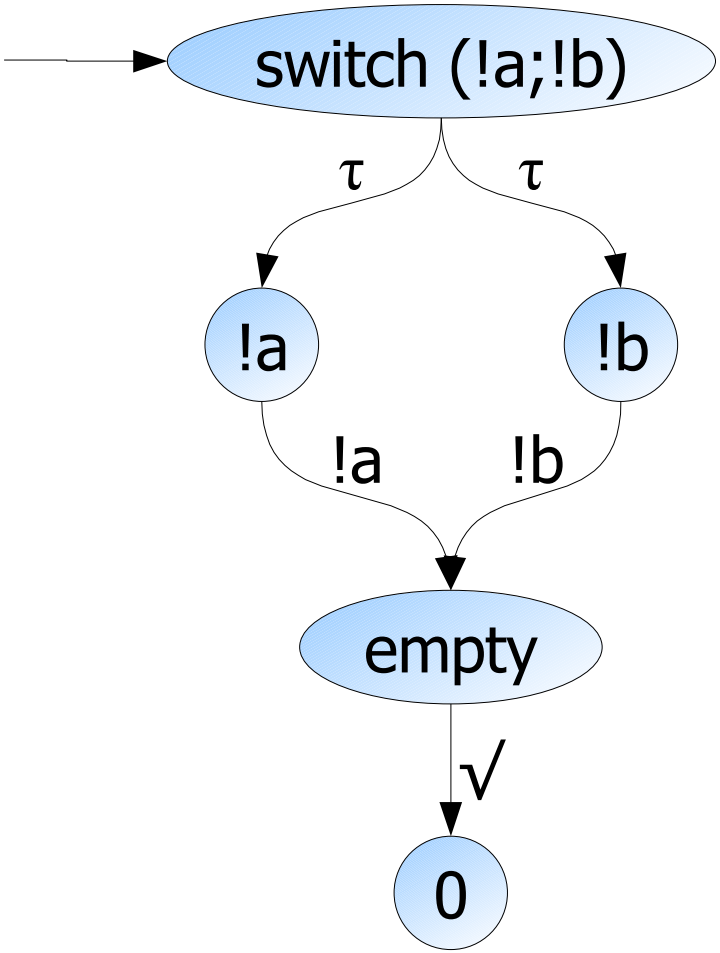
Construction de l'AT du service

(sans horloge)

!a



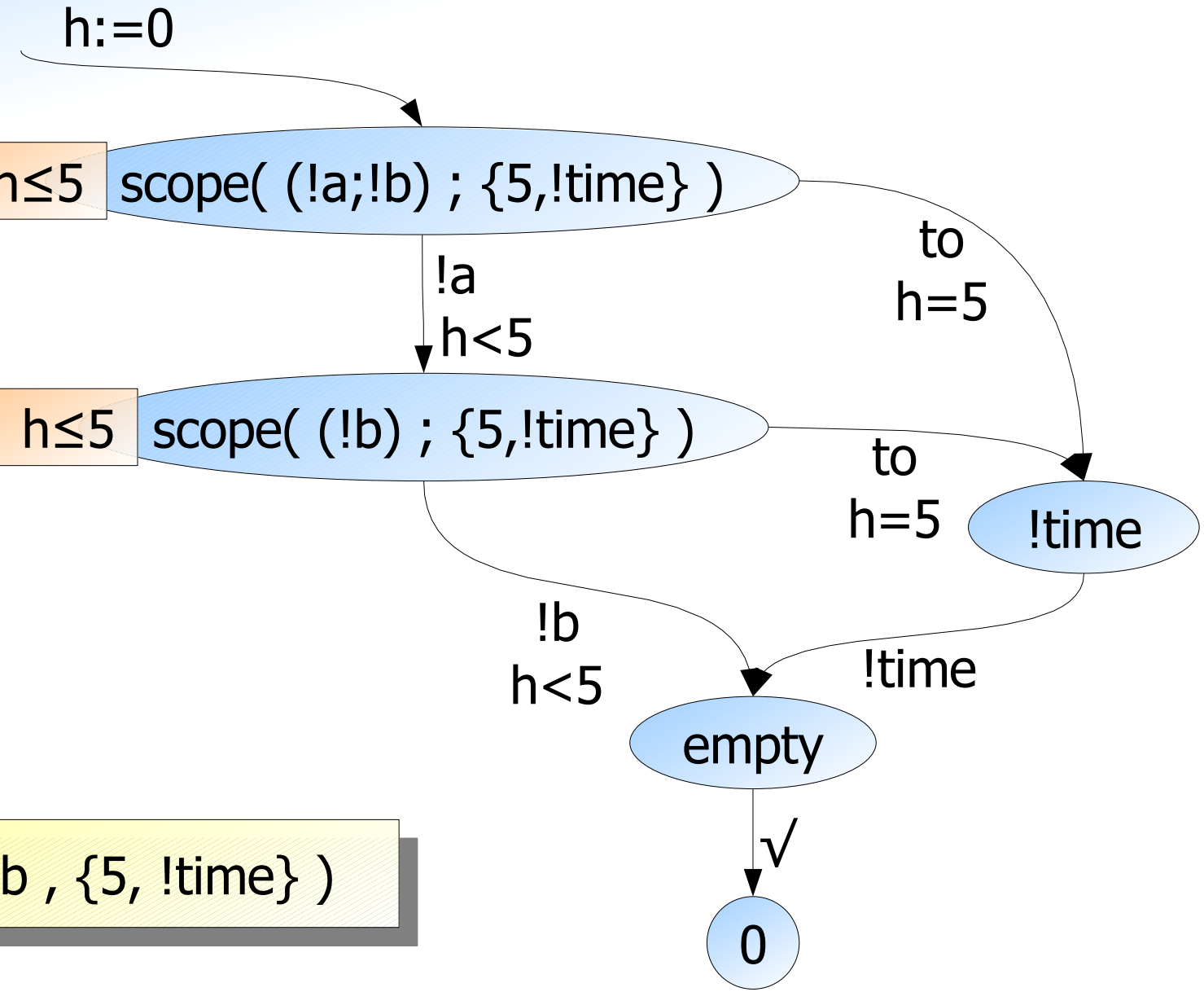
switch(!a;!b)



construction de l'AT : application récursive des règles

Construction de l'AT du service

(avec des horloges)



scope(!a;!b , {5, !time})

Relation d'interaction
et
Génération du client

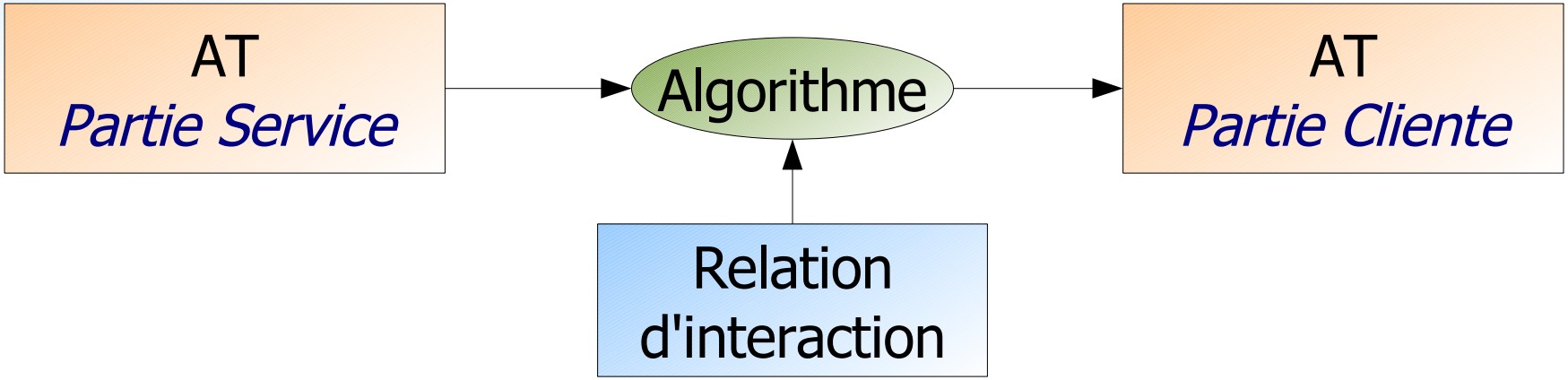
Relation d'interaction

Quel client pour un service ?

- Le client doit être un programme :
  - qui communique par envoi et réception de messages,
  - qui gère le temps,
- donc : représentation par un automate temporisé.
- A tout instant de l'interaction :
  - l'ensemble des messages que le client est susceptible d'envoyer est l'ensemble des messages attendus par le service.
  - tout message attendu par le client correspond à un message susceptible au vu des messages échangés d'être envoyé par le serveur.
- Le client est capable de détecter la terminaison de l'interaction.

Il y a ambiguïté si un tel client n'existe pas

Génération du client



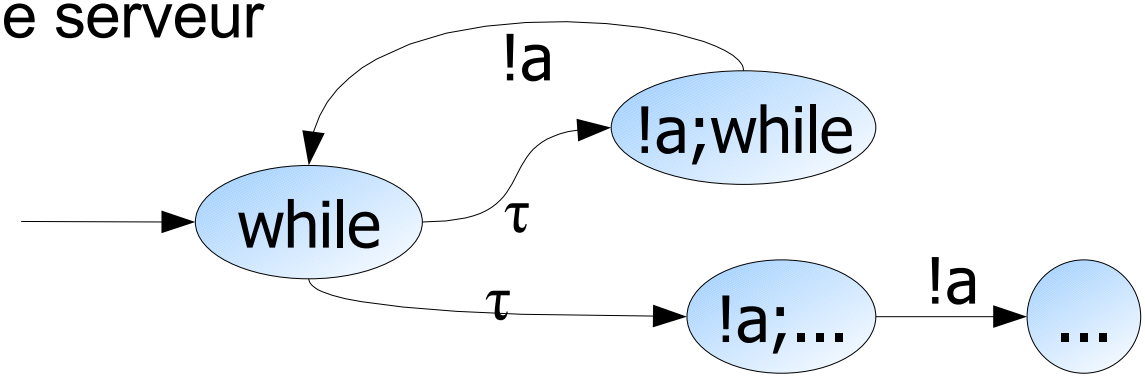
- Le client doit être implémentable : donc il doit s'exécuter de façon déterministe
- Le client doit avoir les mêmes horloges que le service
- L'algorithme de génération du client doit produire un AT qui est en *relation d'interaction* avec le service (échanges de messages et horloges).

Ambiguïté d'un service

exemple (1)

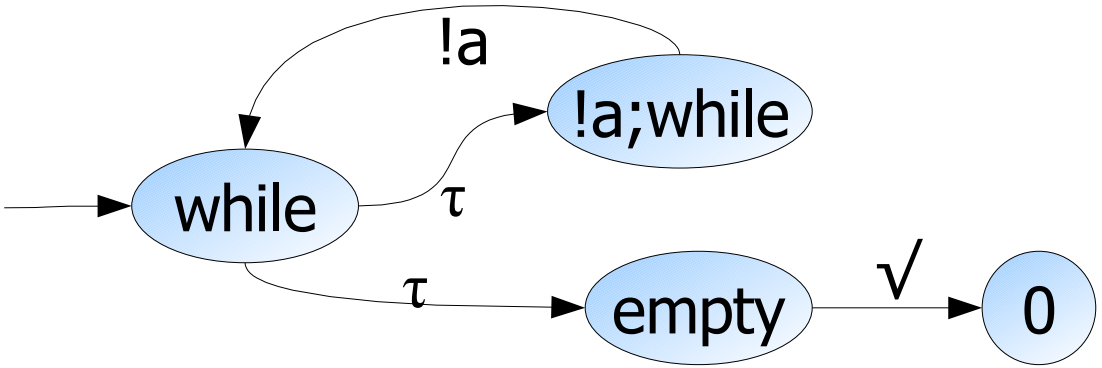
while { !a } ; !a ; ...

- **(sans doute) ambigu :** car après réception du message, le client ne sait pas dans quel état est le serveur



while { !a }

- **ambigu :** car le client ne sait pas combien il recevra de messages avant la fin de l'exécution

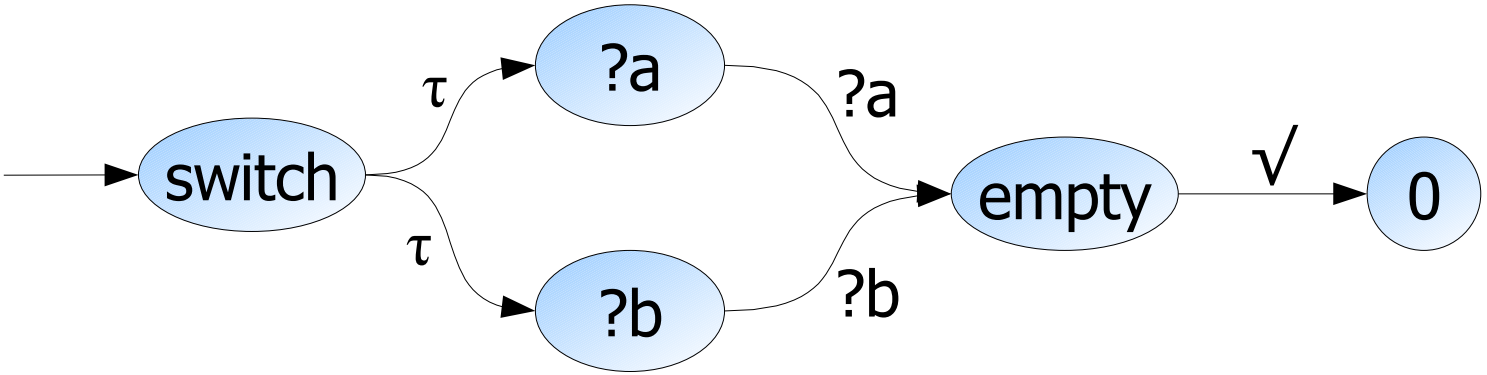


Ambiguïté d'un service

exemple (2)

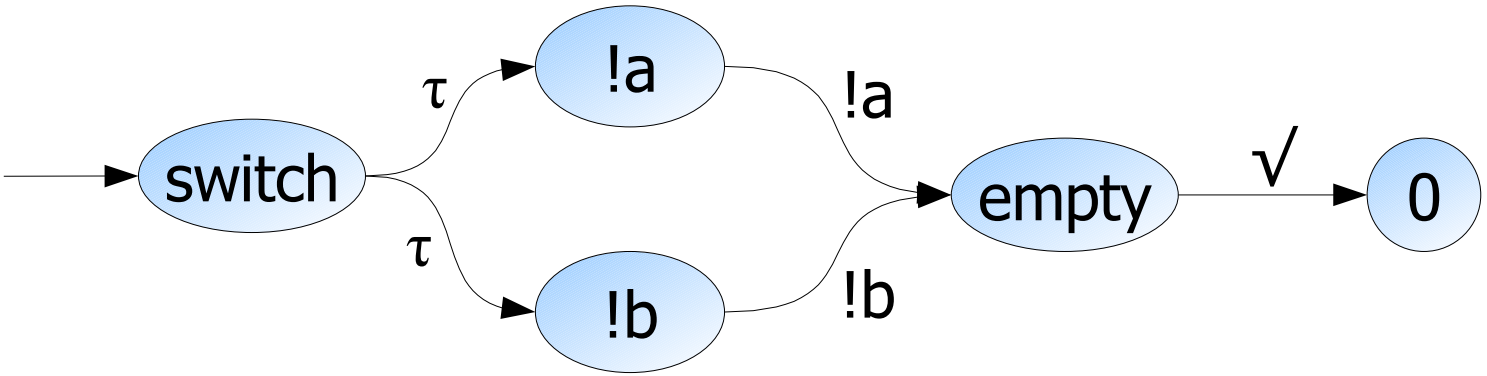
switch { ?a , ?b }

- **ambigu :** car le client ne sait pas quel message envoyer



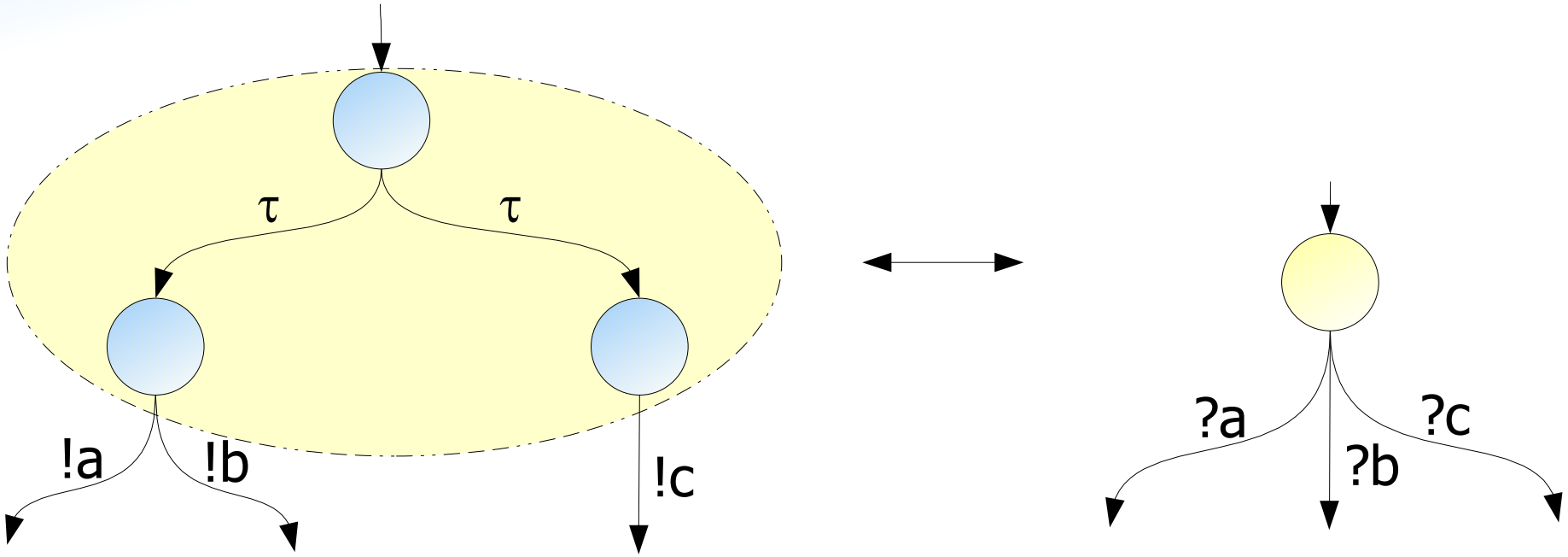
switch { !a , !b }

- **non ambigu :** car le client attendra l'un ou l'autre des messages



Algorithme de génération

agrégation d'états (τ-closure)

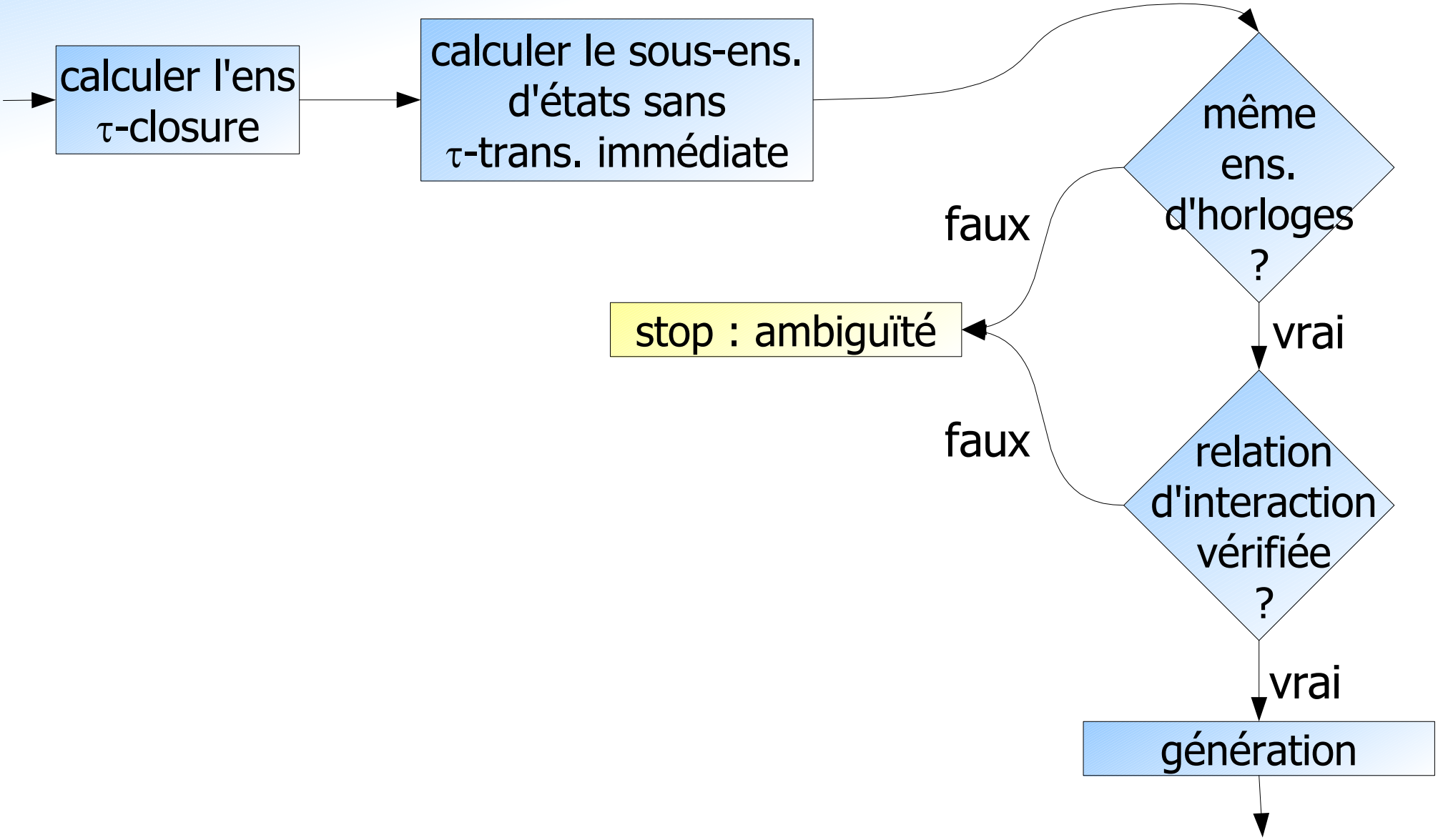


AT Service

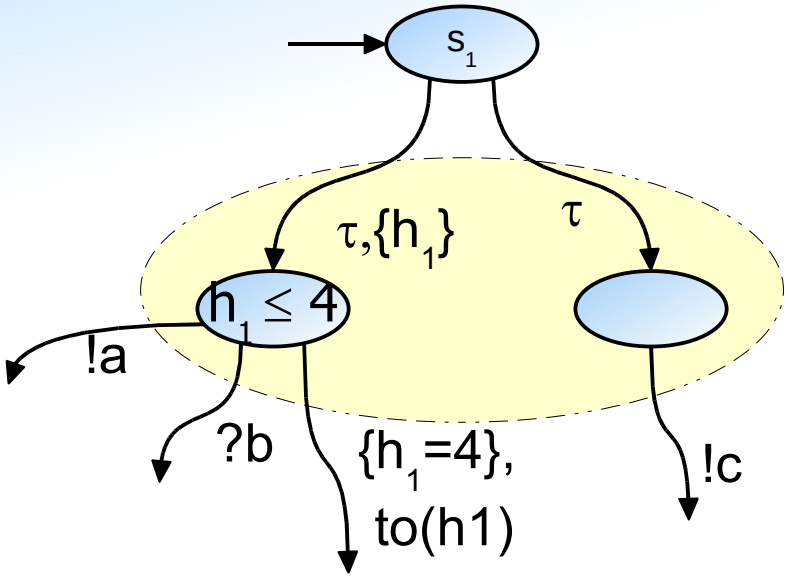
AT Client

Algorithme de génération

les étapes



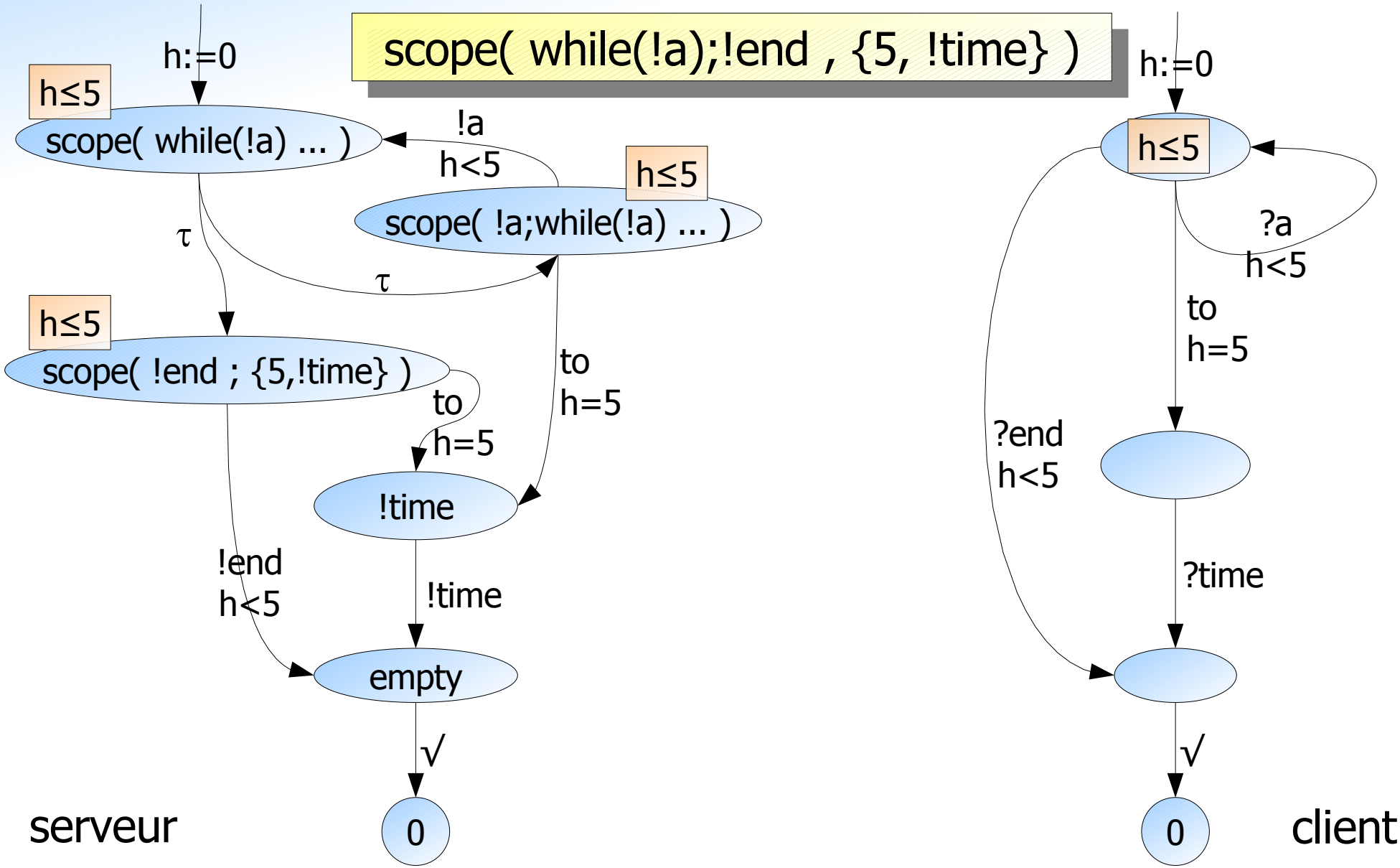
L'algorithme est « incomplet »




```
switch (
  !c,
  scope( !a,
         {?b, empty},
         {4, empty} )
)
```

- La branche *scope* nécessite l'activation d'une horloge. L'autre branche non.
- Le client ne peut savoir s'il doit déclencher l'horloge ou non ! (et à quel moment)
- Conclusion de l'algorithme : Ambiguïté temporelle !
- *En temps discret, ce problème avait une solution...*

Exemples d'automates du service et du client



L'exécution du client
(encore en développement)

- Suivre l'exécution indiquée par l'AT client (obtenu par les algorithmes et à partir d'un fichier de description BPEL quelconque)
- Fonctionnalités :
  - Déclencher les horloges lors d'une transition
  - Vérifier en permanence les transitions encore franchissables (en cas de gardes)
  - Proposer à l'utilisateur un formulaire pour remplir les champs des messages à envoyer au serveur
  - Masquer les formulaires que l'utilisateur n'a pas rempli en temps voulu
  - Gérer l'arrivée et l'envoi des messages

Conclusion et Perspectives

Conclusion

- En temps discret et en temps dense : modélisation de la partie service, puis de la partie cliente.
- Relation d'interaction qui permet de détecter si le service est ambigu
- Génération d'un client générique

- Technologies mouvantes
  - développement d'outils génériques (langages : XLANG/BPEL, règles sémantiques)

Perspectives

- Partie client générique : exécution du client
- Ajout des caractéristiques du canal de communication dans les différents algorithmes
- Intégration à la plate-forme du projet « Services Web » au LAMSADE

Bibliographie

[HMMR04a] *A Dense Time Semantics for Web Services Specifications Languages* S. Haddad, T. Melliti, P. Moreaux, and S. Rampacek (ICTTA'04)

[HMMR04b] *Modelling Web Services Interoperability* S. Haddad, T. Melliti, P. Moreaux, and S. Rampacek (ICEIS04)

[HMR06] *Client synthesis for Web Services by way of a timed semantics* S. Haddad, P. Moreaux, and S. Rampacek (ICEIS06)