Composition Dynamique de Fonctionnalités de Dispositifs en Réseau dans le Web Sémantique

Sattisvar TANDABANY

LIG – HADAS

Directeur de thèse: Marie-Christine ROUSSET
Rapporteur : Djamal BENSLIMANE
Rapporteur : Farouk TOUMANI
Président du jury : Christine COLLET

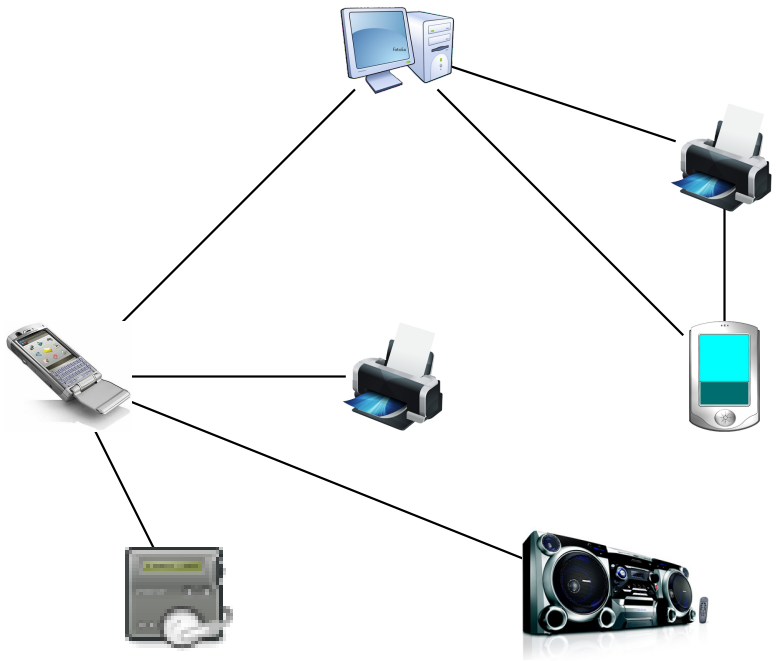
23 novembre 2009

LIG

HADAS

UJF

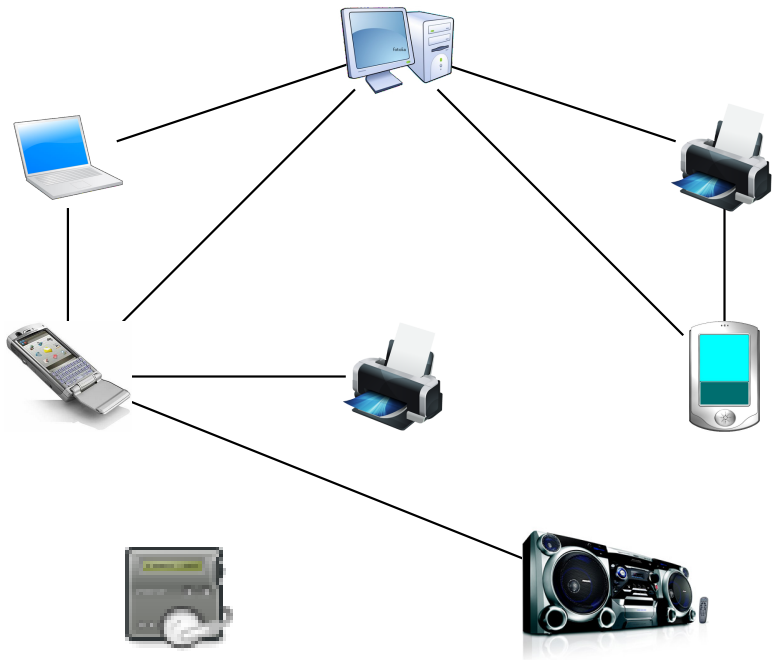
Contexte



Informatique Ambiante

- Réseaux opportunistes de dispositifs
- Sans infrastructure
- Offre de fonctionnalités par les dispositifs

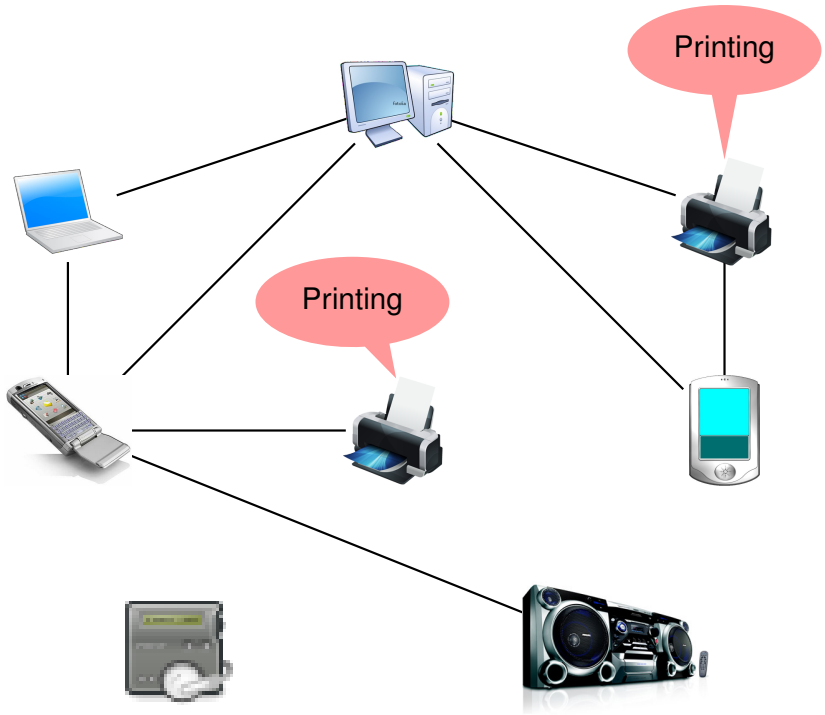
Contexte



Informatique Ambiante

- Réseaux opportunistes de dispositifs
- Sans infrastructure
- Offre de fonctionnalités par les dispositifs

Contexte



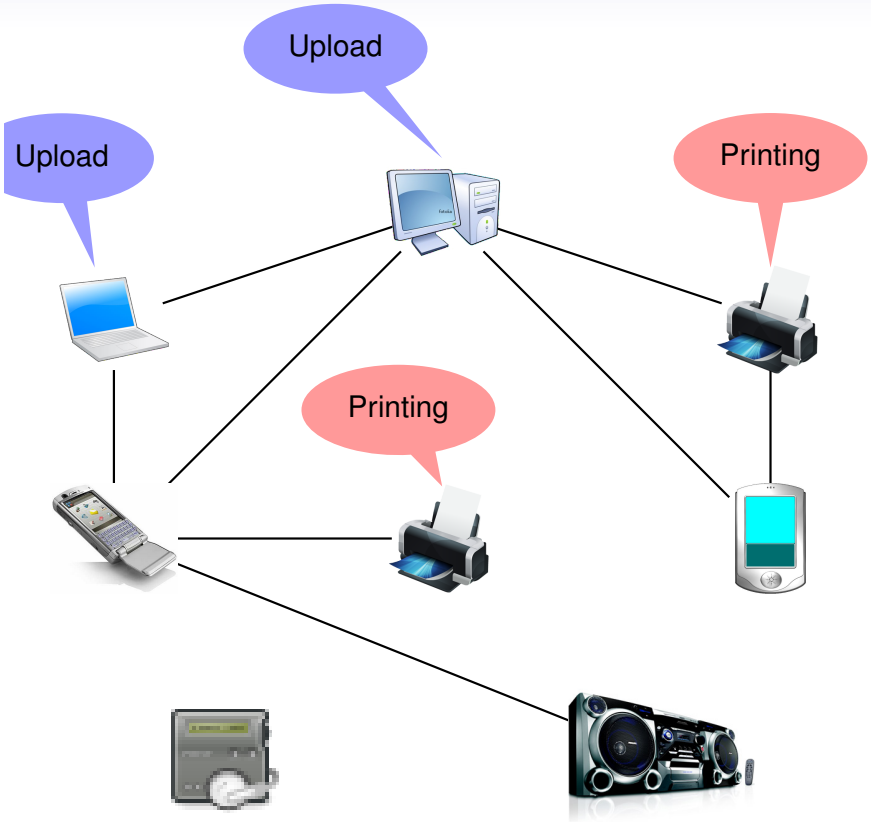
Informatique Ambiante

- Réseaux opportunistes de dispositifs
- Sans infrastructure
- Offre de fonctionnalités par les dispositifs

Exemples

- Impression

Contexte



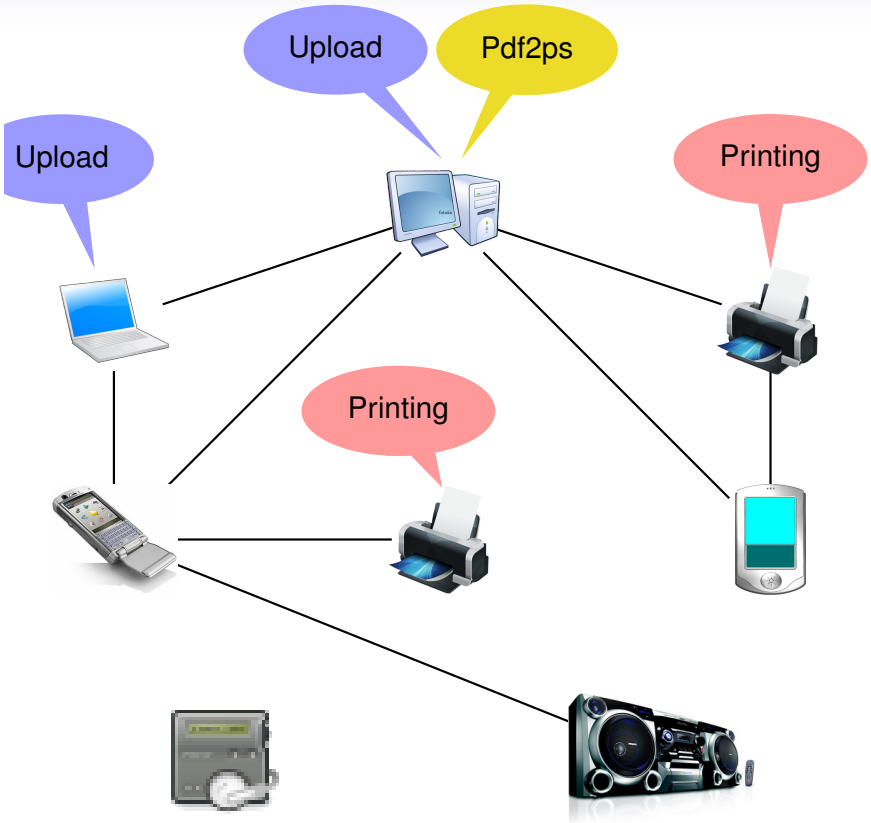
Informatique Ambiante

- Réseaux opportunistes de dispositifs
- Sans infrastructure
- Offre de fonctionnalités par les dispositifs

Exemples

- Impression
- Téléchargement

Contexte



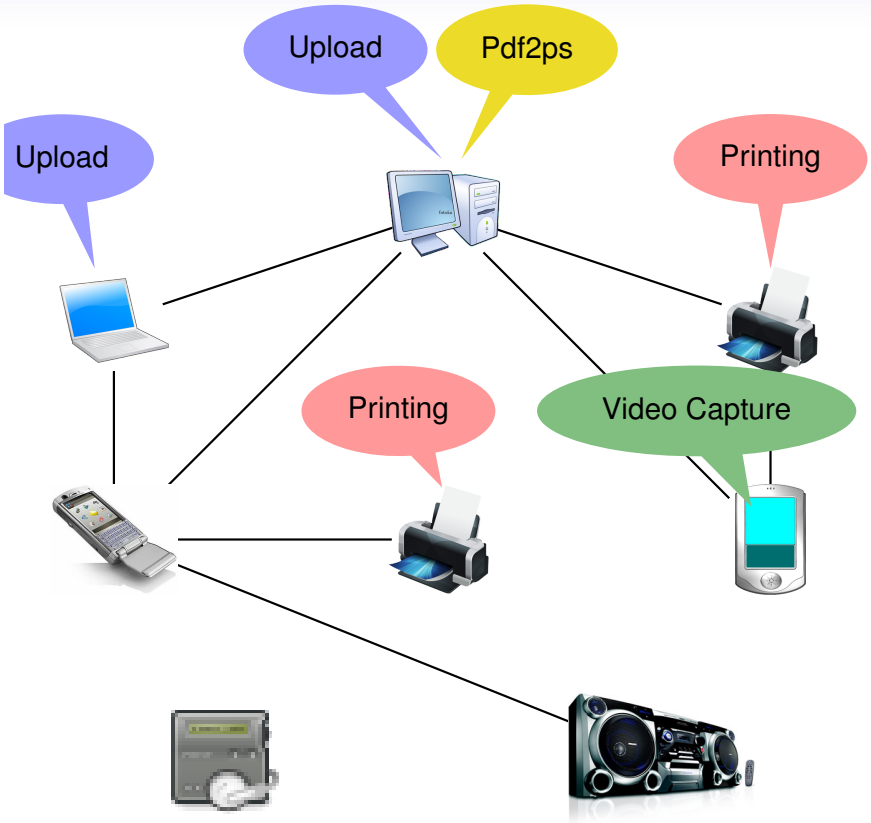
Informatique Ambiante

- Réseaux opportunistes de dispositifs
- Sans infrastructure
- Offre de fonctionnalités par les dispositifs

Exemples

- Impression
- Téléchargement
- Conversion pdf ps

Contexte



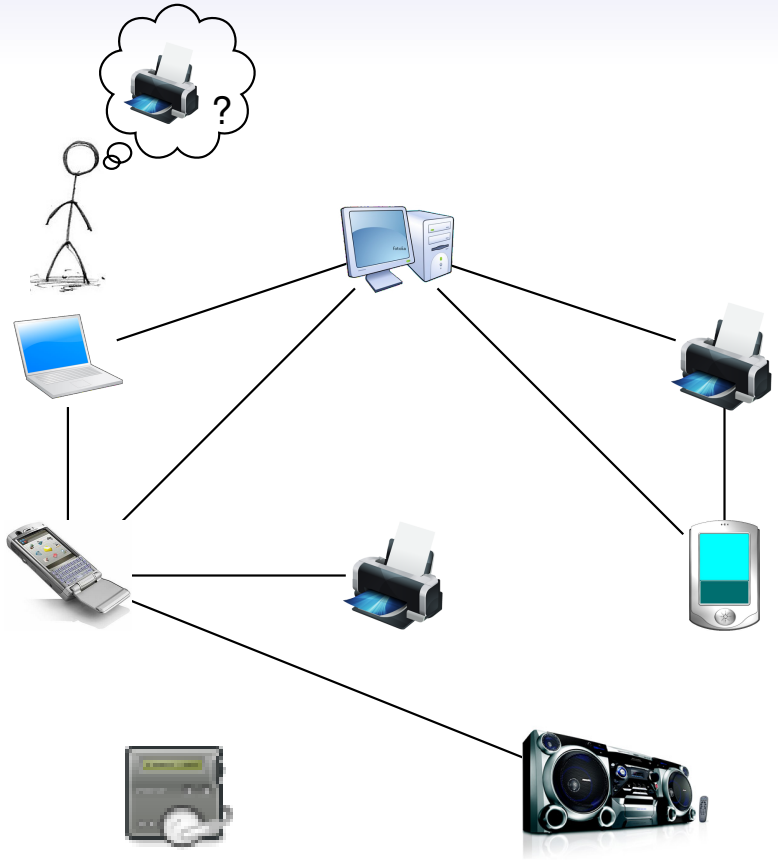
Informatique Ambiante

- Réseaux opportunistes de dispositifs
- Sans infrastructure
- Offre de fonctionnalités par les dispositifs

Exemples

- Impression
- Téléchargement
- Conversion pdf ps
- Capture vidéo

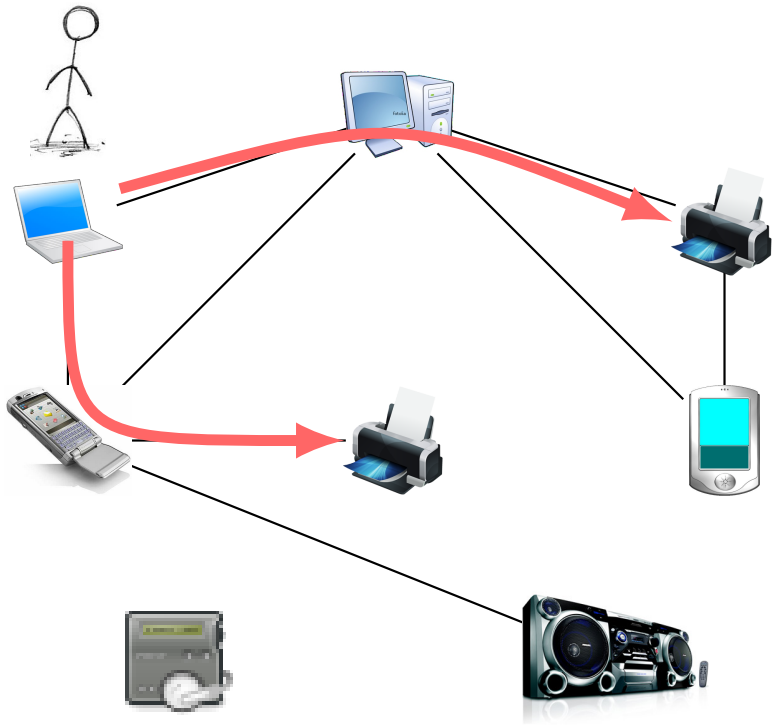
Motivations



Spécifier un besoin

- langage de haut niveau
- fonctionnalité complexe

Motivations



Spécifier un besoin

- langage de haut niveau
- fonctionnalité complexe

Répondre au besoin

- par compositions de fonctionnalités offerte
- automatiquement et à la demande

Problématique

- L'absence d'infrastructure rend difficile la recherche de dispositifs et de fonctionnalités
- Le caractère dynamique du réseau nécessite de construire les compositions à la volée

Étant données :

- la description d'une requête
- la description des dispositifs connectés au réseau

La problèmatique est de

1. **Trouver** parmi tous les dispositifs, ceux qui sont **pertinents** pour la requête
2. **Construire** les compositions des fonctionnalités de ces dispositifs qui **satisfont** la requête

Travaux connexes dans l'état de l'art

- Peu de travaux sur les dispositifs leurs fonctionnalités et leurs compositions
- La composition de services Web est très proche de nos travaux

Comparaison avec les compositions de services Web

Abstraction

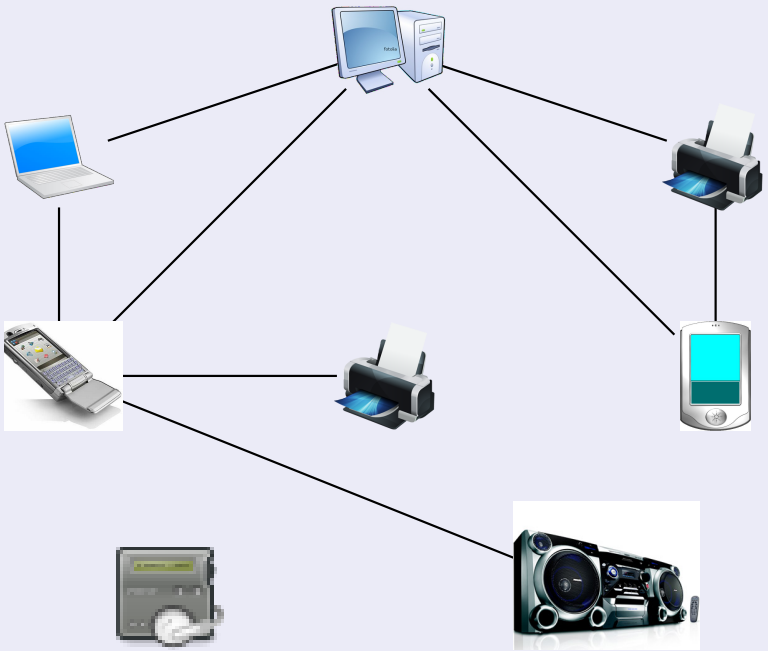
- Dispositifs = Services Web
- Fonctionnalités = Méthodes

3 approches représentatives

- par instanciation de motifs précalculés
- par un médiateur sémantique dédié
- par le calcul situationnel

Travaux connexes dans l'état de l'art (1)

Composition par application de motifs



pdf2ps → download → printing

upload → pdf2ps → printing

video → transfer → display

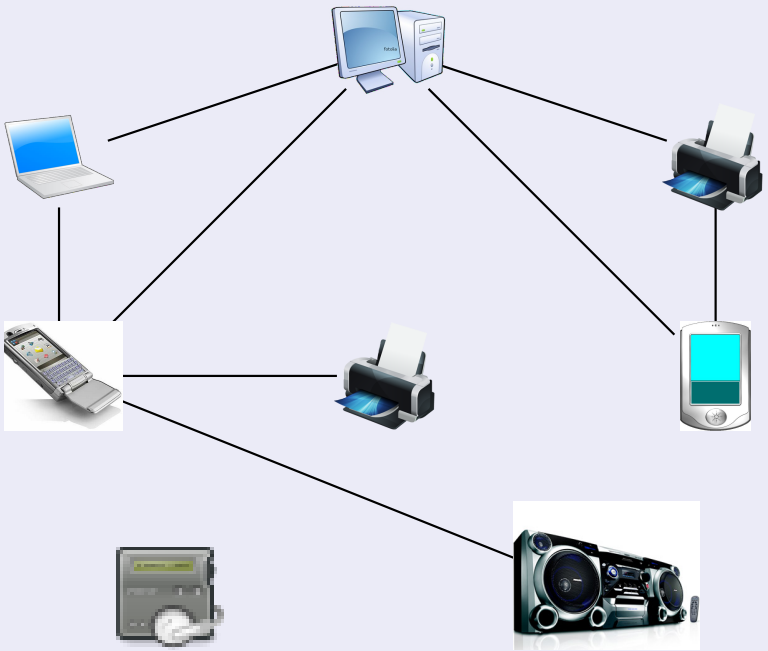
play music → equalizer → speaker

Problèmes

- La composition est précalculée et instanciée à la demande
- Capacité des dispositifs dépassée par la quantité de motifs à retenir

Travaux connexes dans l'état de l'art (1)

Composition par application de motifs



pdf2ps → download → printing

video → transfer → display

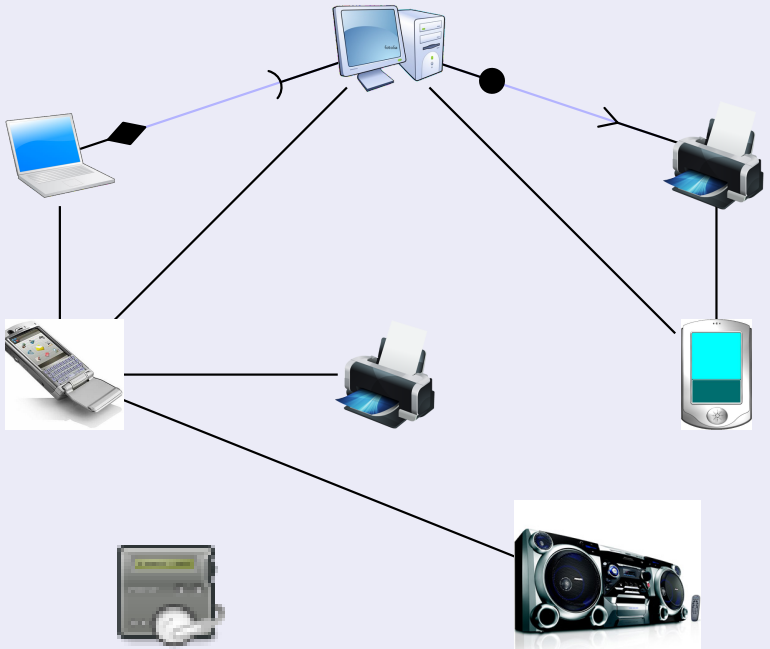
play music → equalizer → speaker

Problèmes

- La composition est précalculée et instanciée à la demande
- Capacité des dispositifs dépassée par la quantité de motifs à retenir

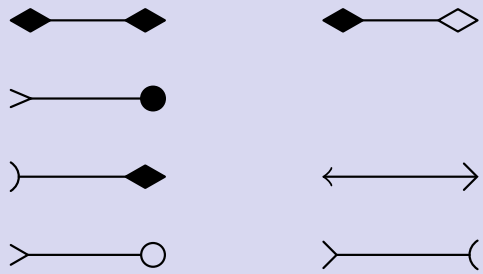
Travaux connexes dans l'état de l'art (2)

Composition à l'aide de médiateur sémantique externe



Wiederhold[3], Haller[4], IRS-III[5], ...

Médiateur sémantique



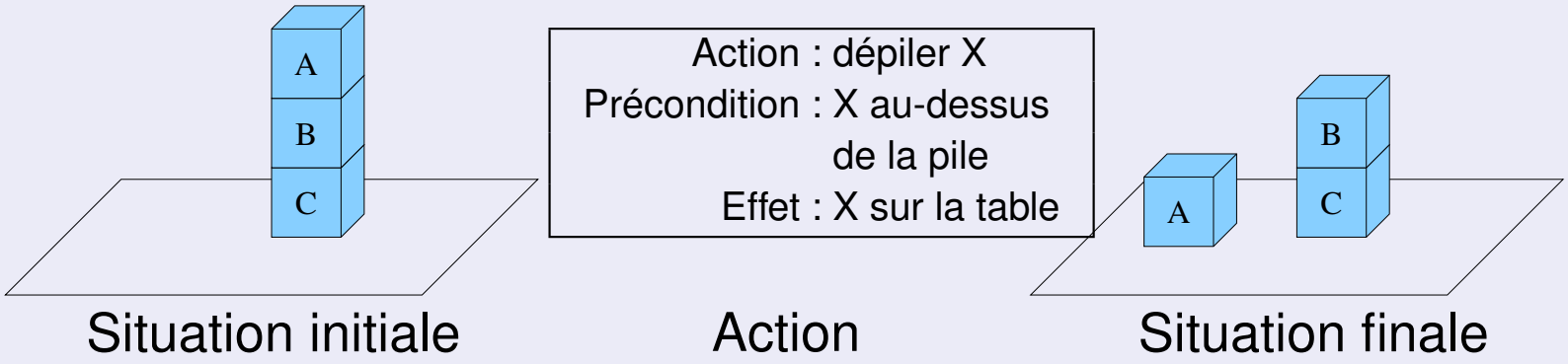
Problèmes

- Difficile d'avoir un point fixe atteignable
- Le médiateur sémantique ne peut pas être exhaustif dans un réseau dynamique

Travaux connexes dans l'état de l'art (3)

Calcul situationnel

Situation : État du monde
Action : Méthode
Requête : situation cible
Réponse : succession d'actions pour atteindre la situation cible



McCarthy[6], McIlraith[7], ...

Problème

Exprimer le transfert de données entre fonctionnalités nécessite une extension

Notre approche

Une approche basée sur la logique et le raisonnement

- Pouvoir d'expression de la logique
- Cadre formel pour prouver des propriétés (complétude/correction/...)

Notre approche

Une approche basée sur la logique et le raisonnement

- Pouvoir d'expression de la logique
- Cadre formel pour prouver des propriétés (complétude/correction/...)

Un langage logique déclaratif

- En logique des prédicats

pour :

- décrire des **dispositifs**
- décrire des **fonctionnalités**
  fonctions reliant entrées et sorties
- spécifier des **requêtes**

Notre approche

Une approche basée sur la logique et le raisonnement

- Pouvoir d'expression de la logique
- Cadre formel pour prouver des propriétés (complétude/correction/...)

Un langage logique déclaratif

- En logique des prédicats

pour :

- décrire des **dispositifs**
- décrire des **fonctionnalités**
  fonctions reliant entrées et sorties
- spécifier des **requêtes**

Un moteur de raisonnement

- Chaînage arrière (Prolog)

pour :

- construire les **compositions de fonctionnalités**
  logique des prédicats (fonctions) nécessaire
- découvrir les **dispositifs pertinents**
  logique propositionnelle suffisante

Contributions

Définition d'un langage de description des dispositifs et des requêtes basé sur la logique

Implémentation et validation expérimentale du calcul des compositions de fonctionnalités

Exploitation de SOMEWHERE[1] pour la découverte des dispositifs pertinents

Plan

Langage de descriptions

Les descriptions sont fournies par l'utilisateur en langage de haut niveau

Le langage contient :

- Une taxonomie de classes
- Des prédicats mettant en relations des constantes du langages
- La fonction *out*
- Des faits décrivants les dispositifs
- Des règles logiques décrivant les fonctionnalités

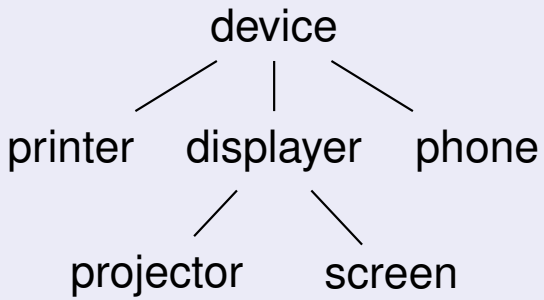
Le langage ne contient pas :

- de négation

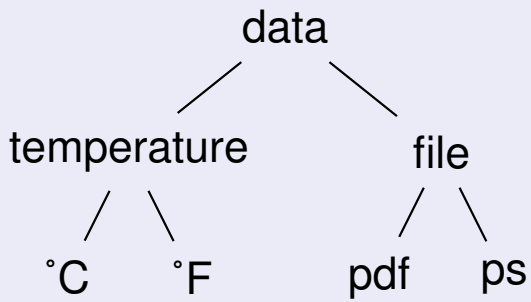
Langage : les taxonomies

C 1

Classes de dispositifs



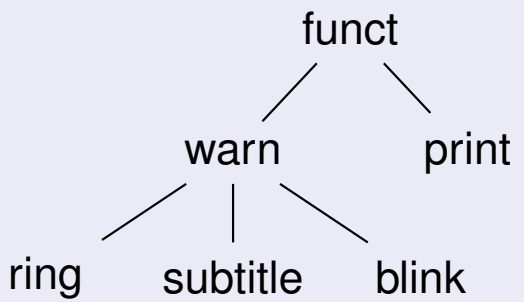
Classes de données



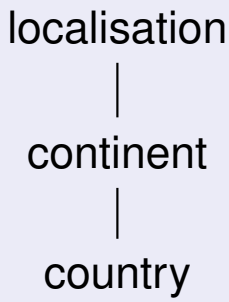
Intérêts

- typage
- exprimable avec RDFS[2]
- facilité d'enrichissement

Classes de fonctionnalité



Classes de localisation

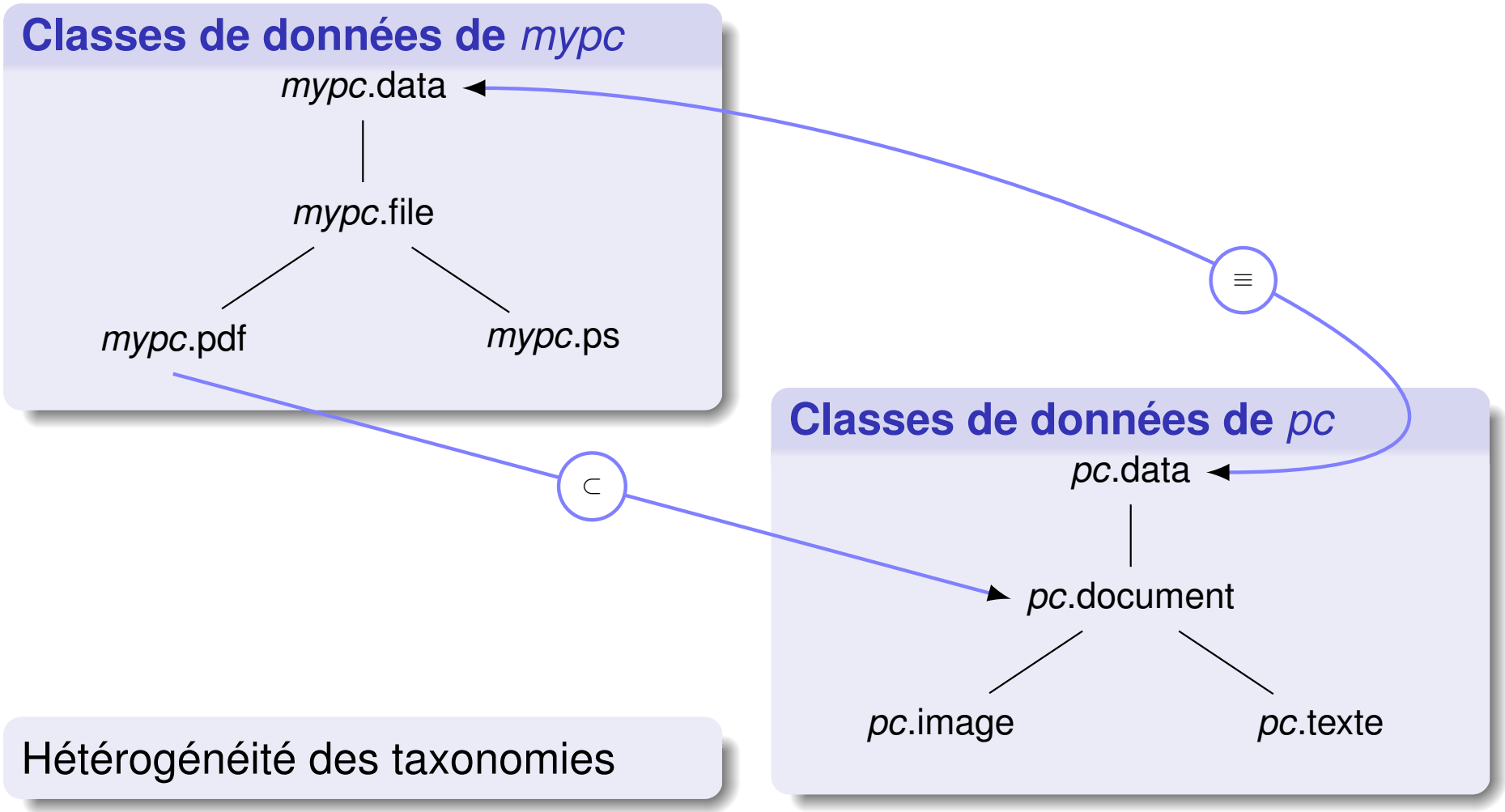


Règles logique

- Relation de sous-classe : des faits
- Héritage du typage : 1 règle

Langage : correspondance entre taxonomies

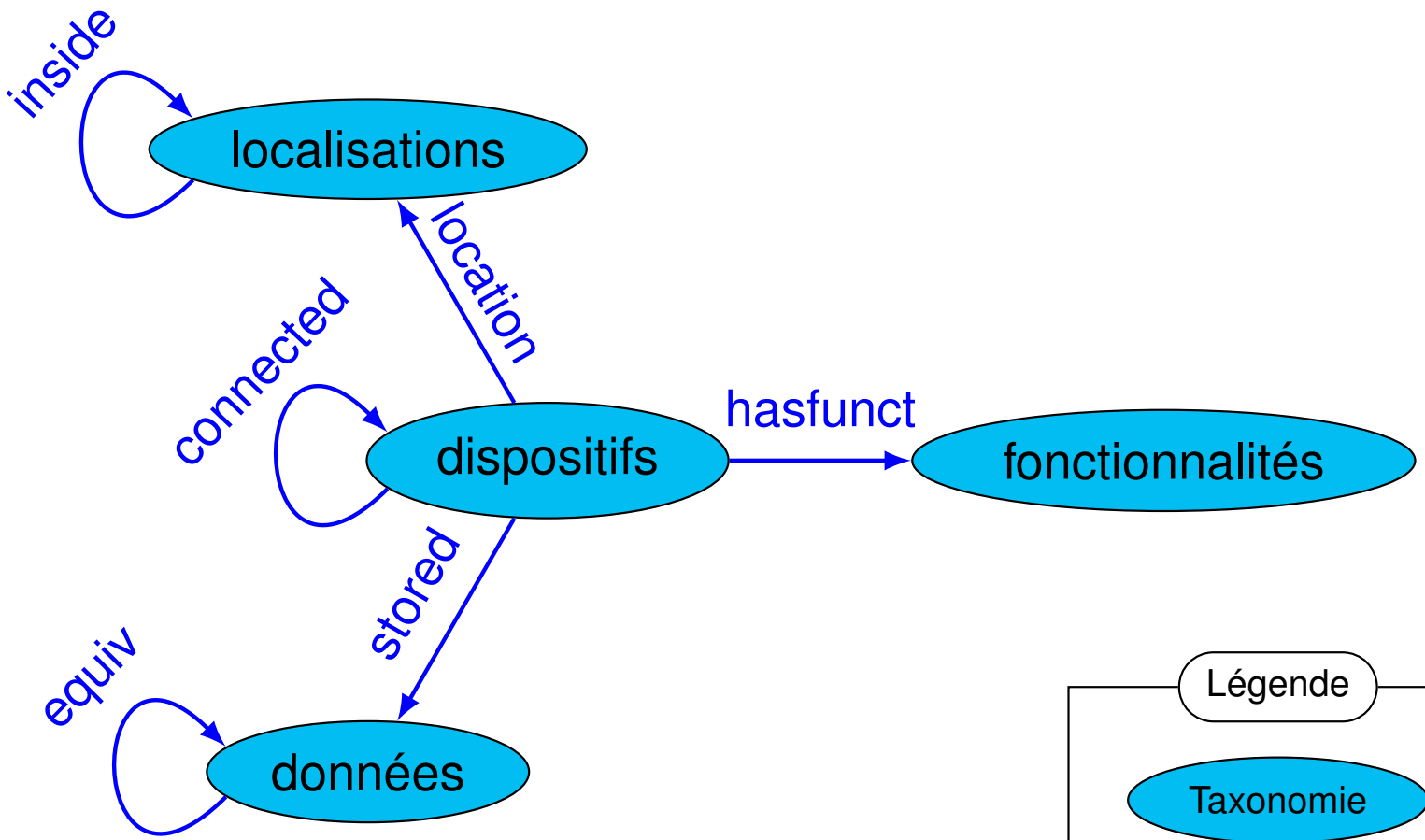
C 1



Hétérogénéité des taxonomies

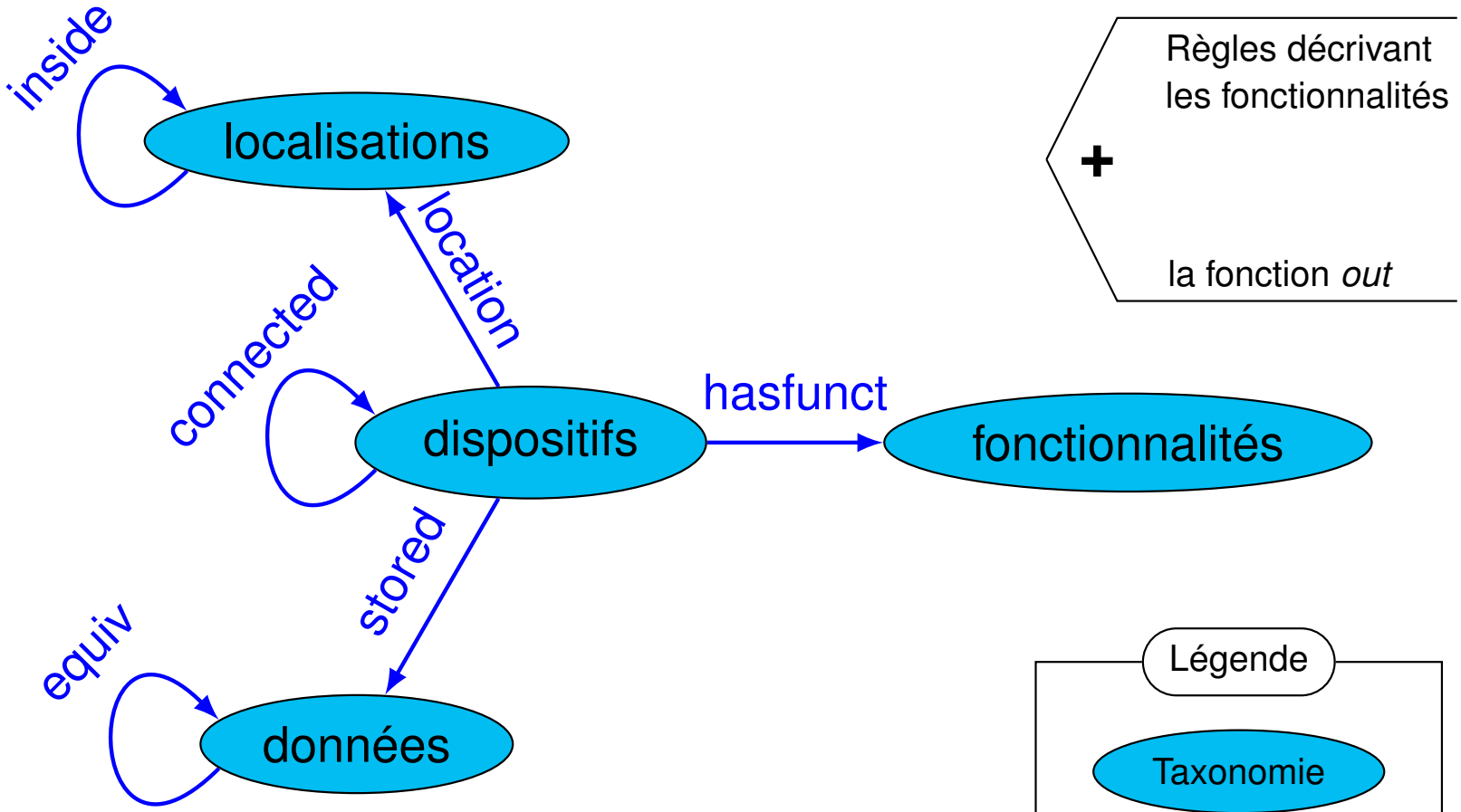
Langage : les prédicats

C 1



Langage : les prédicats

C 1



Langage : les fonctionnalités

C 1

Une fonctionnalité est vue comme une fonction

Caractéristiques

- attendent *des données en entrée*
- fournissent *une donnée en sortie*
- nécessitent parfois des *préconditions*
- assurent des *post-conditions*

La fonctionnalité *pdf2ps* de l'ordinateur *pc*

- hasfunct(*pc*, pdf2ps(*pc*))
- type(pdf2ps(*pc*), *pdf2ps*)
- type(I, *pdf*) $\wedge$ type(F, *pdf2ps*) $\wedge$ hasfunct(E, F) $\wedge$ stored(I, D) $\wedge$ connected(D, E)
  $\rightarrow$ type(out(F,[I]),*ps*) $\wedge$ stored(out(F,[I]),E) $\wedge$ equiv(out(F,[I]),I)

Langage : La fonction *out*

C 1

La fonction *out*

Soit F une fonctionnalité, on exprime la sortie de cette fonctionnalité en fonction de ses entrées

$$out(F, [I_1, \ldots, I_n])$$

représente la sortie de la fonctionnalité F avec les entrées $I_1, \ldots, I_n$

Intérêts

- Lien entre la sortie et les entrées
- Propriétés de la sortie
- Cette fonction est la clef de la composition

Plan

Exemple de description de fonctionnalités

La fonctionnalité *pdf2ps* de l'ordinateur *pc*

1. hasfunct(*pc*, pdf2ps(*pc*))
2. type(pdf2ps(*pc*), *pdf2ps*)
3. type(I, *pdf*) $\wedge$ type(F, *pdf2ps*) $\wedge$ hasfunct(E, F) $\wedge$ stored(I, D) $\wedge$ connected(D, E)
   $\rightarrow$ type(out(F,[I]),*ps*) $\wedge$ stored(out(F,[I]),E) $\wedge$ equiv(out(F,[I]),I)

Interprétation

1. *pc* possède la fonctionnalité pdf2ps
2. la fonctionnalité pdf2ps est typée
3. Règle définissant la fonctionnalité
   - Type de l'entrée
   - Type de la sortie
   - Préconditions
   - Post-conditions

Exemple de description de dispositifs

L'ordinateur *pc*

1. type(*pc*, *computer*)
2. hasfunct(*pc*, pdf2ps(*pc*))
3. connected(*pc*, *prn*)

Interprétation

1. *pc* est un ordinateur
2. *pc* possède la fonctionnalité pdf2ps
3. *pc* est connecté à *prn*

Plan

Formalisation de l'approche

Descriptions des dispositifs

- D l'ensemble des dispositifs du réseau
- $\forall d \in D \quad Descr(d)$ la description du dispositif d
- $\Delta = \bigcup_{d \in D} Descr(d)$

Requêtes

$$Def(Q) : \bigwedge_{i=1}^{n} P_i \rightarrow Q(X_0, \ldots, X_p)$$

Réponses

$$Answer(Q, \Delta) = \left\{ \bar{t} \in H(\Delta)^p \quad / \quad \Delta, Def(Q) \models Q(\bar{t}) \right\}$$

Plan

Exemple de requête et sa réponse

Description

mypdf est stocké dans *mypc*
mypc est connecté à *pc*
pc est connecté à *prn* ....

Traducteur

Requête, en langage naturel

obtenir une sortie papier dans la salle S du fichier mypdf stocké sur mon ordinateur

Traducteur

Exemple de requête et sa réponse

Description

mypdf est stocké dans *mypc*
mypc est connecté à *pc*
pc est connecté à *prn* ....

Traducteur

Requête, en langage naturel

obtenir une sortie papier dans la salle S du fichier mypdf stocké sur mon ordinateur

Traducteur

Sachant que

stored(mypdf, mypc)
connected(mypc, pc)
connected(pc, prn) ....

Raisonneur

Requête, expression logique

type(**X**,paper) $\wedge$ **X**=out(F,[I]) $\wedge$
equiv(I,mypdf) $\wedge$ hasfunct(D,F) $\wedge$
located(D,roomS) $\longrightarrow$ Q(**X**)

Exemple de requête et sa réponse

Description

mypdf est stocké dans *mypc*
mypc est connecté à *pc*
pc est connecté à *prn* ....

Traducteur

Requête, en langage naturel

obtenir une sortie papier dans la salle S du fichier mypdf stocké sur mon ordinateur

Traducteur

Sachant que

stored(mypdf, mypc)
connected(mypc, pc)
connected(pc, prn) ....

Raisonneur

Requête, expression logique

type(**X**,paper) $\wedge$ **X**=out(F,[I]) $\wedge$
equiv(I,mypdf) $\wedge$ hasfunct(D,F) $\wedge$
located(D,roomS) $\longrightarrow$ Q(**X**)

Réponse de Prolog : instanciation de la variable d'intérêt *X*

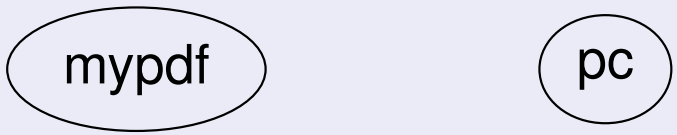
$$X =$$
$$out(printing(prn), [out(pdf2ps(pc), [out(upload(mypc), [mypdf, pc])])])$$

Lecture de la composition dans la réponse

Réponse de Prolog

$out(printing(prn), [out(pdf2ps(pc), [out(upload(mypc), [mypdf, pc])])])$

Arbre de composition

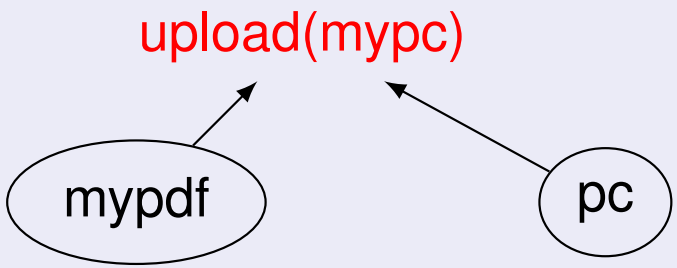


Lecture de la composition dans la réponse

Réponse de Prolog

$out(printing(prn), [out(pdf2ps(pc), [out(upload(mypc), [mypdf, pc])])])$

Arbre de composition

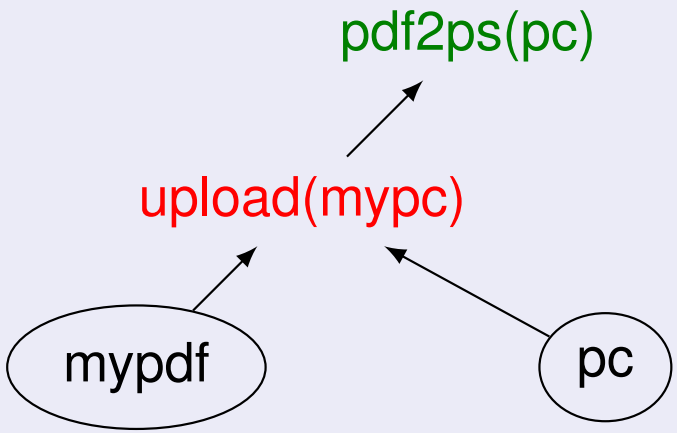


Lecture de la composition dans la réponse

Réponse de Prolog

$out(printing(prn), [out(pdf2ps(pc), [out(upload(mypc), [mypdf, pc])])])$

Arbre de composition

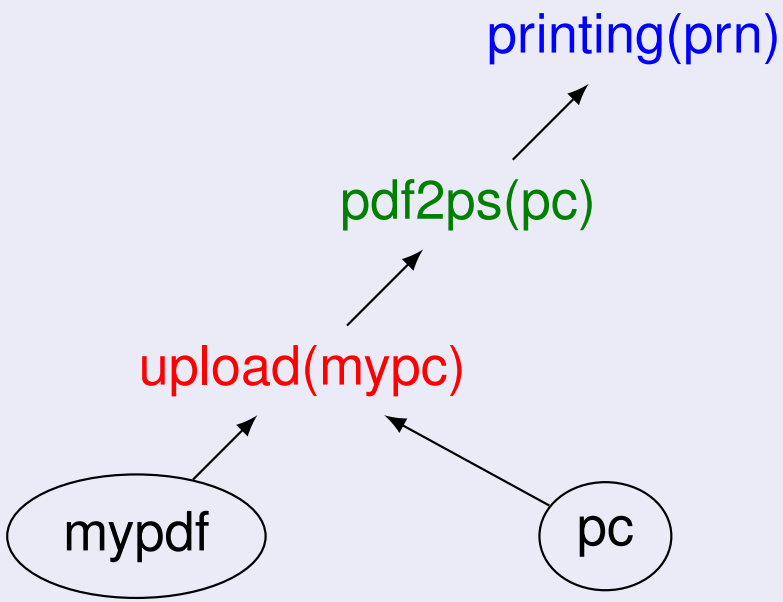


Lecture de la composition dans la réponse

Réponse de Prolog

$out(printing(prn), [out(pdf2ps(pc), [out(upload(mypc), [mypdf, pc])])])$

Arbre de composition



Plan

Choix du raisonneur

C 2

Prolog

- Chaînage arrière
- Logique des prédicats (dont les fonctions)

XSB Prolog

- Utilisation de table indexée de prédicats déjà prouvés
- Permet d'éviter un type de boucles infinies

Exemple de boucle évitable

- $equiv(X, Y) \rightarrow equiv(Y, X)$
- $equiv(a, b)$.

$equiv(b, c)$? Non

Exemple de boucle inévitable

- $type(I, b) \rightarrow type(out(f1, [I]), a)$
- $type(J, a) \rightarrow type(out(f2, [J]), b)$
- $type(X, a) \rightarrow q(X)$

$q(X)$?
$X = out(f1, [out(f2, [out(f1, [...])])])$

Protocole expérimental

C 2

Matériel

- XSB Prolog
- Simple Processeur Intel Xeon 2.68GHz, 4Gb RAM, 32bits

5 Scénarios de simulation

- n imprimantes
- n serveurs
- 1 client
- 1 composition de profondeur 2
- 1 composition de profondeur n
- 1 comp. de prof. n sans types

Paramètres

- nombre de dispositifs $(2n+1)$
- type et nombre de connexions

Mesures

- Le temps de calcul
- L'espace mémoire utilisé

Scénario : Meilleur cas

C 2

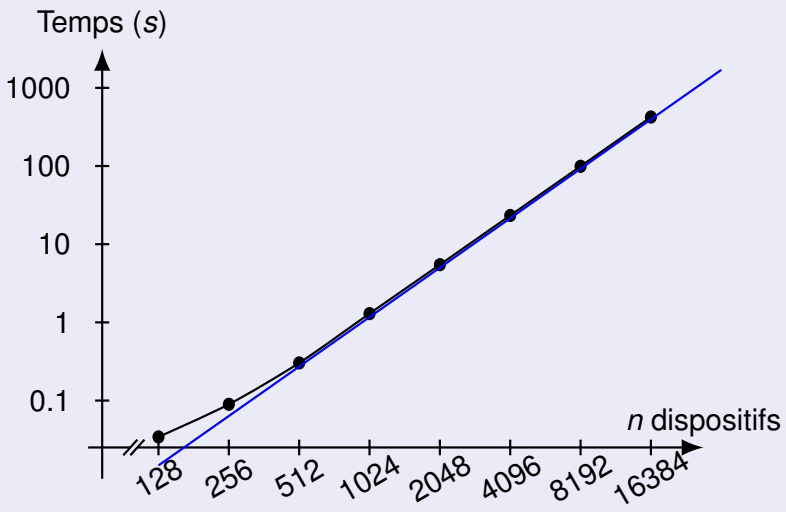
1 composition de profondeur 2

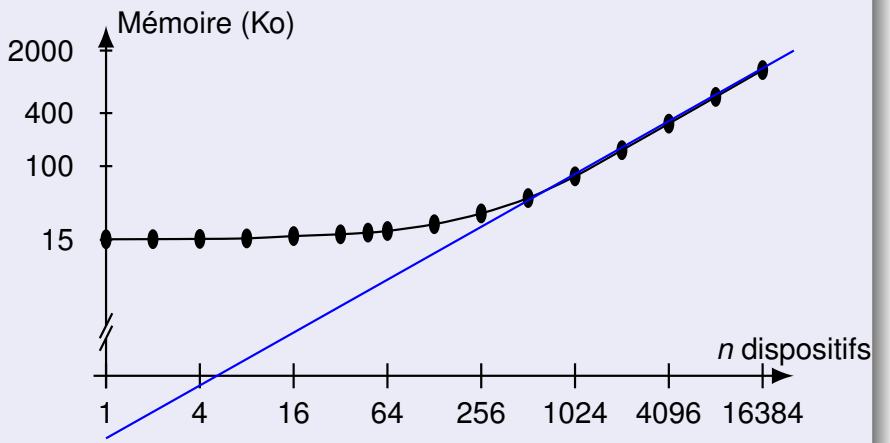
$\texttt{mypc}$ — $\texttt{srv}_1$ — $\texttt{prn}_1$

$\texttt{srv}_i$ $\texttt{prn}_i$

$\texttt{srv}_n$ $\texttt{prn}_n$

Résultats expérimentaux

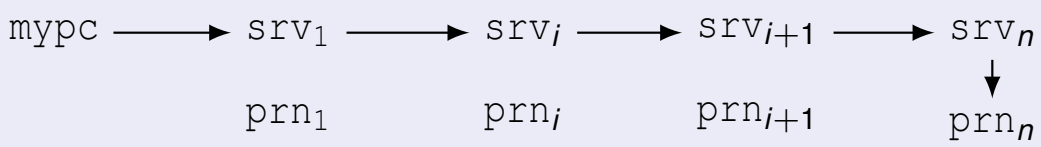




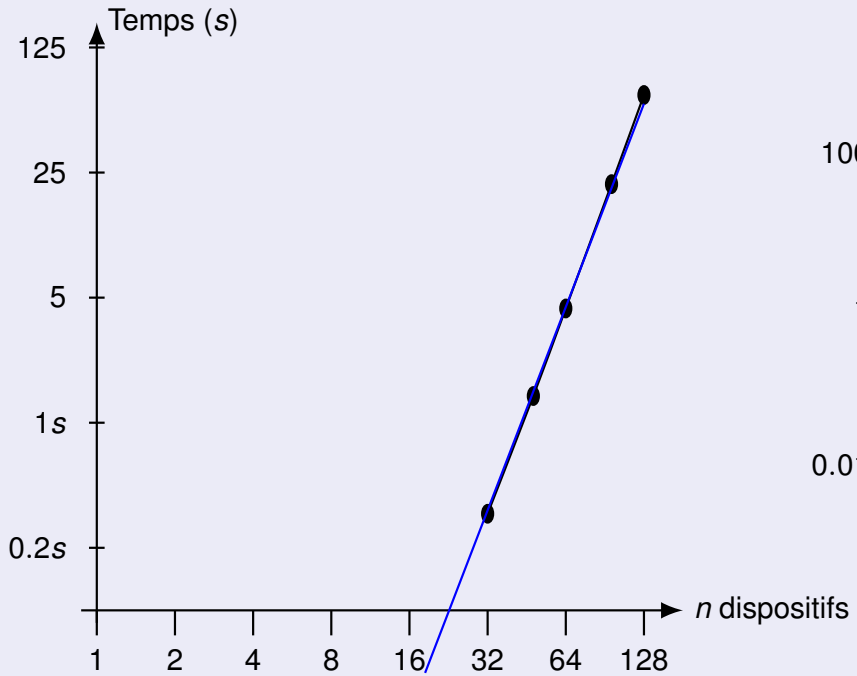
Scénarios (2)

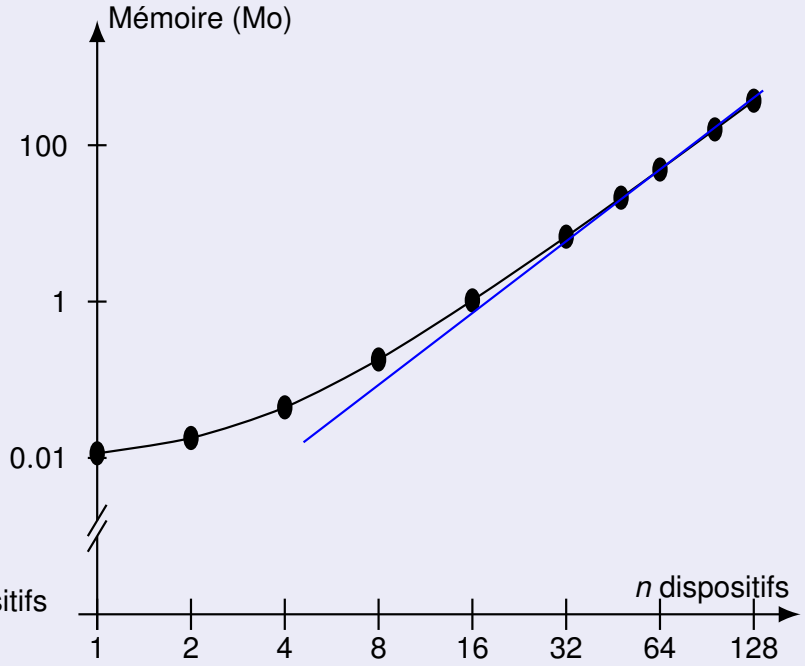
C 2

1 composition de profondeur *n*



Résultats expérimentaux



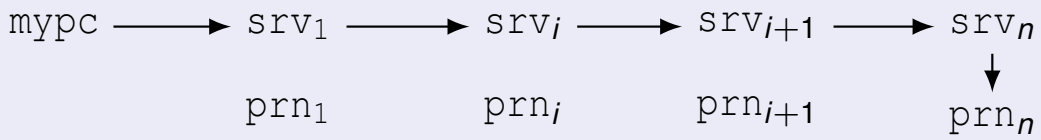


Scénarios (3)

C 2

1 composition de profondeur *n*

Sans exploitation des types



Résultats expérimentaux

- Temps d'exécution exponentiel avec le nombre de dispositifs
- Utilisation mémoire exponentielle avec le nombre de dispositifs

Synthèse des expérimentations

C 2

De l'intérêt du typage

- Sans exploitation du typage, le temps de calcul est exponentiel
- L'exploitation des taxonomies permet de réduire l'espace de recherche

Les dispositifs parasites

- Les dispositifs parasites n'apparaissent pas dans les réponses,
- Trouver les dispositifs pertinents permet d'éviter ce problème

Récapitulatif des scénarios

Compositions		Mesures	
Nombre	Profondeur	Temps	Mémoire
1	2	n^2	n
1	n	n^4	n^3
n	2	n^3	n^2
n^2	2	n^4	n^2
2^n	n	$n^2 2^n$	$n^2 2^n$

Plan

Définitions des dispositifs pertinents à une requête

Nécessaire au calcul d'une réponse

- Q une requête
- d un dispositif
- d est un dispositif pertinent ssi

$$Answer\left(Q,\ \Delta - Descr\left(d\right)\right) \neq Answer\left(Q,\ \Delta\right)$$

- d est nécessaire au calcul de la requête Q

SomeWhere

Outils

- Raisonneur distribué de la logique des propositions
- Peut être déployé sur les dispositifs (code Java)

Algorithme

- termine
- passe à l'échelle
- est correct
- est complet

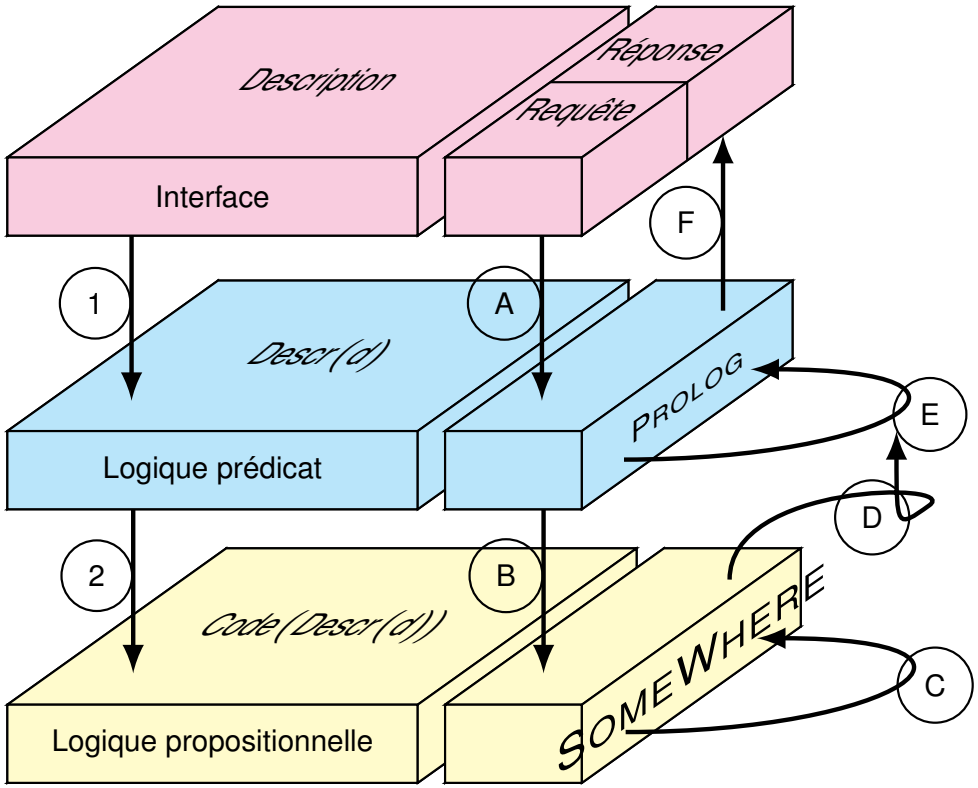
Utiliser pour trouver tous les dispositifs pertinents

- Encodage des descriptions en logique propositionnelle
- Encodage des requêtes en logique propositionnelle
- SomeWhere retourne une liste de tous les dispositifs pertinents pour la requête

Détail de l'approche

Définition

$$Lookup(Q) = \bigcup_{P \in Ppi^{\star}(Code(Q),\Delta)} Decode(P)$$



Théorème de transfert

C 3

Théorème

Pour une requête donnée, on obtient exactement les mêmes réponses en exécutant Prolog sur l'ensemble des descriptions qu'en l'exécutant sur le sous ensemble des descriptions des dispositifs pertinents

$$Answer\,(Q,\ \Delta) = Answer\,(Q,\ Lookup\,(Q))$$

Esquisse de la preuve

Mise en correspondance des chemins succès dans l'arbre de dérivation d'une requête en logique des prédicats et les chemins succès dans l'arbre de dérivation d'une requête en logique propositionnelle

Plan

Conclusions

Langage Logique déclaratif

- Un langage pour spécifier des demandes de fonctionnalités (composées)
- Un langage pour décrire les dispositifs et les fonctionnalités
- En vue de l'exploitation de Prolog pour le calcul des compositions

Découvrir tous les dispositifs pertinents pour une requête

- Encodage des descriptions et des requêtes
- En vue de l'exploitation de SomeWhere

Calculer les compositions de fonctionnalités répondant à une requête

- Utilisation d'un moteur de raisonnement
- Expérimentation validant l'approche dans le cas où le raisonnement se fait sur l'ensemble des descriptions

Perspectives

- Étudier dans quels cas l'approche utilisée pour découvrir les dispositifs pertinents est optimale et comment l'améliorer dans les autres cas
- Introduire la négation en contournant le problème de Prolog de la négation par l'échec
  - Reporter l'évaluation des termes négatifs en attendant que les toutes variables qui s'y trouvent soit instanciés
- Étendre SomeWhere à la logique des prédicats

Merci de votre attention

Des questions ?

Annexes

A Scénarios (4)

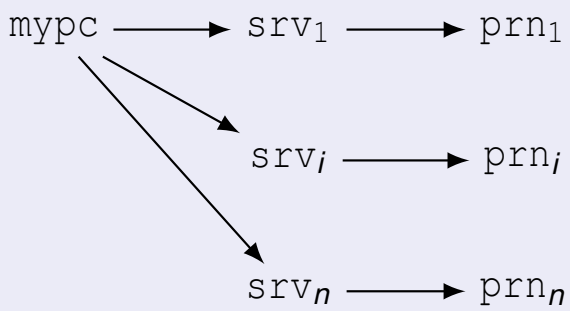
B Scénarios (5)

C Négation par l'échec en Prolog

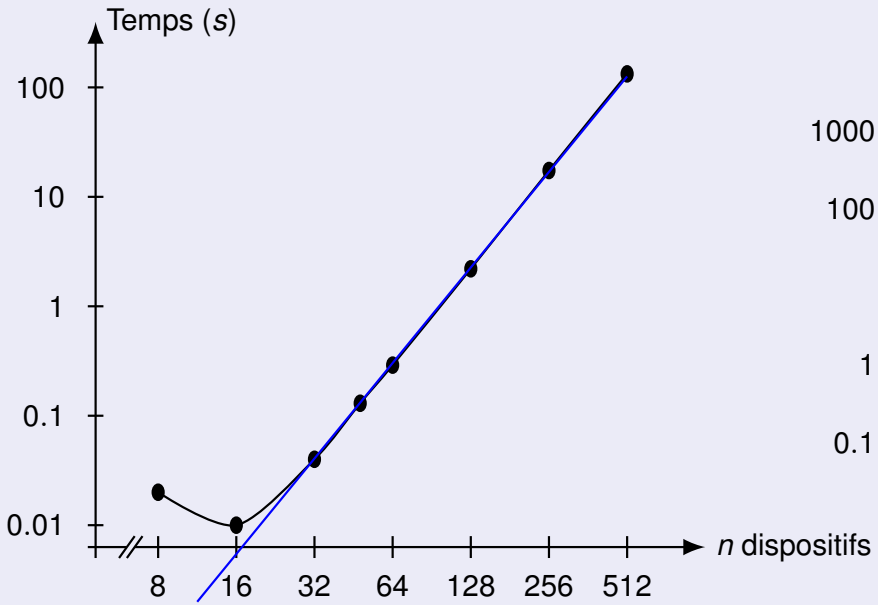
D Bibliographie

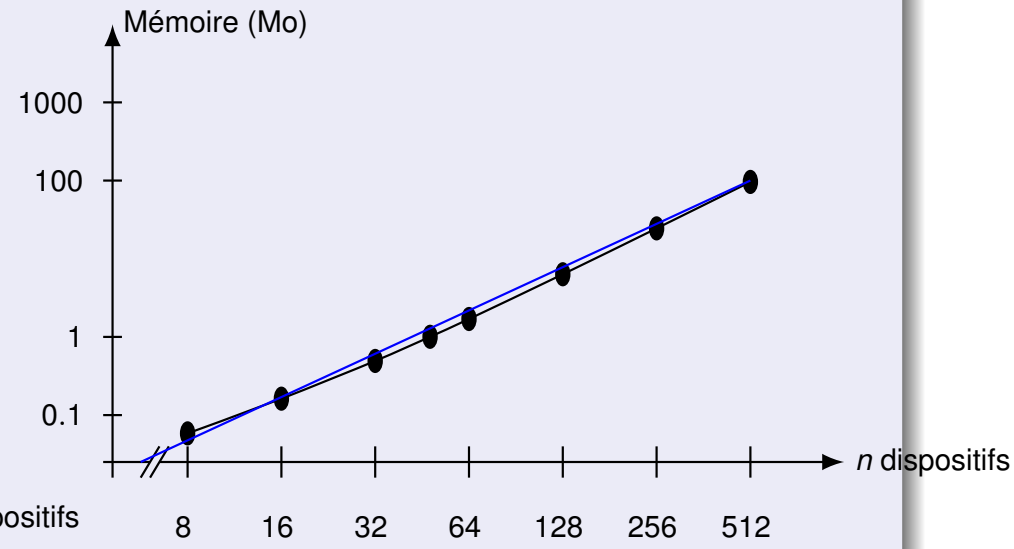
Scénarios (4)

n composition de profondeur 2



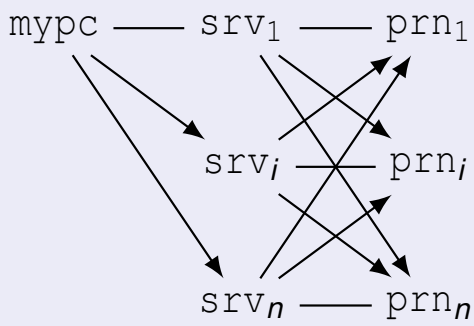
Résultats expérimentaux



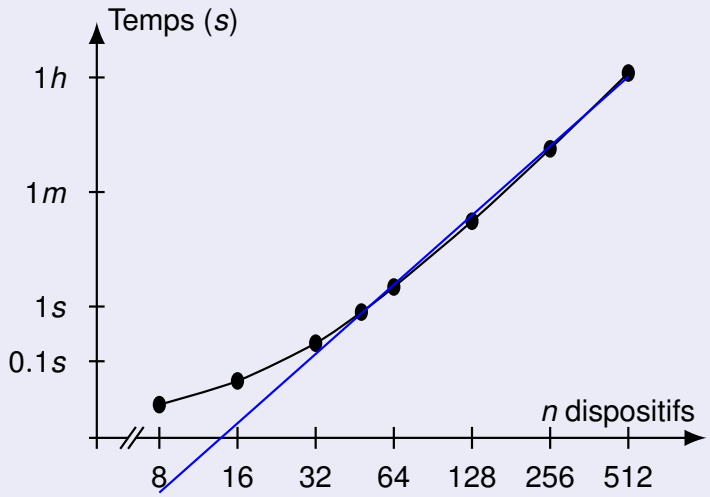


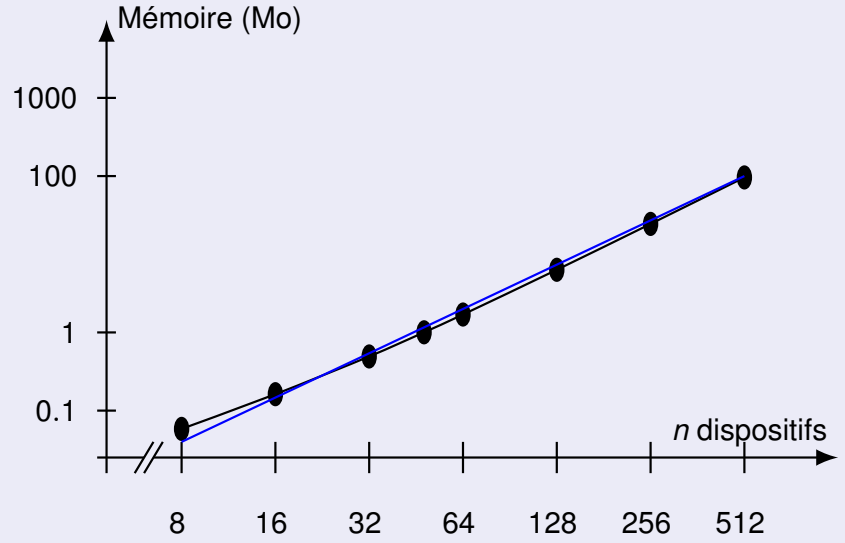
Scénarios (5)

n^2 composition de profondeur 2



Résultats expérimentaux





Négation par l'échec en Prolog

Exemple

- $homme(bob)$
- $femme(X) :- \neg homme(X)$

Requête : femme(alice) ? Oui
Requête : femme(X) ? Non

Bibliographie

1 P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. *Scalability study of peer-to-peer consequence finding*, IJCAI, pages 351—356. 2005.

2 urlhttp ://www.w3.org/TR/rdf-schema/

3 G. Wiederhold. *Mediators in the architecture of future information systems*. IEEE Computer, 25(3) :38—49, 1992.

4 A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. Wsmx – *a semantic service-oriented architecture*. ICWS, pages 321—328. IEEE Computer Society, 2005.

5 L. Cabral and J. Domingue. *Mediation of semantic web services in irs-iii*. In MEDIATE 2005 (ICSOC 2005)

6 J. McCarthy. *Situations, actions and causal laws*. Semantic Information Processing, pages 410—417, 1968.

7 S. McIlraith and T. C. Son. *Adapting golog for composition of semantic web services*. In Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002), pages 482-–493, 2002.