Traitement de requêtes de bon sens sur les actions pour l'interaction homme-machine

Nicolas Sabouret
&
Jean-Paul Sansonnet

Groupe AMI

LIMSI-CNRS

Projet InterViews

MFI'03 - 21 mai 2003

http://www.limsi.fr/Individu/nico

Plan de l'exposé

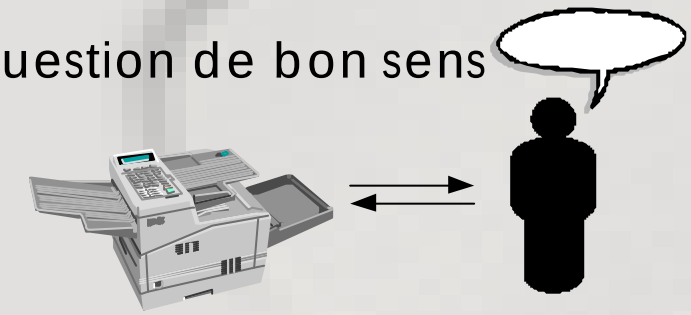
1. Présentation de la problématique
2. Le projet *InterViews*
3. Un modèle de requêtes
4. Manipulation des connaissances de bon sens
5. Un exemple détaillé
6. Conclusion et perspectives

Problématique

Communication Homme-Machine

Utilisateur ordinaire en situation d'échec

Question de bon sens



ex: intelligence ambiante

Intelligence Artificielle

Raisonnement sur les actions

- Planification
- Diagnostic
- Explications
- Common sense reasoning

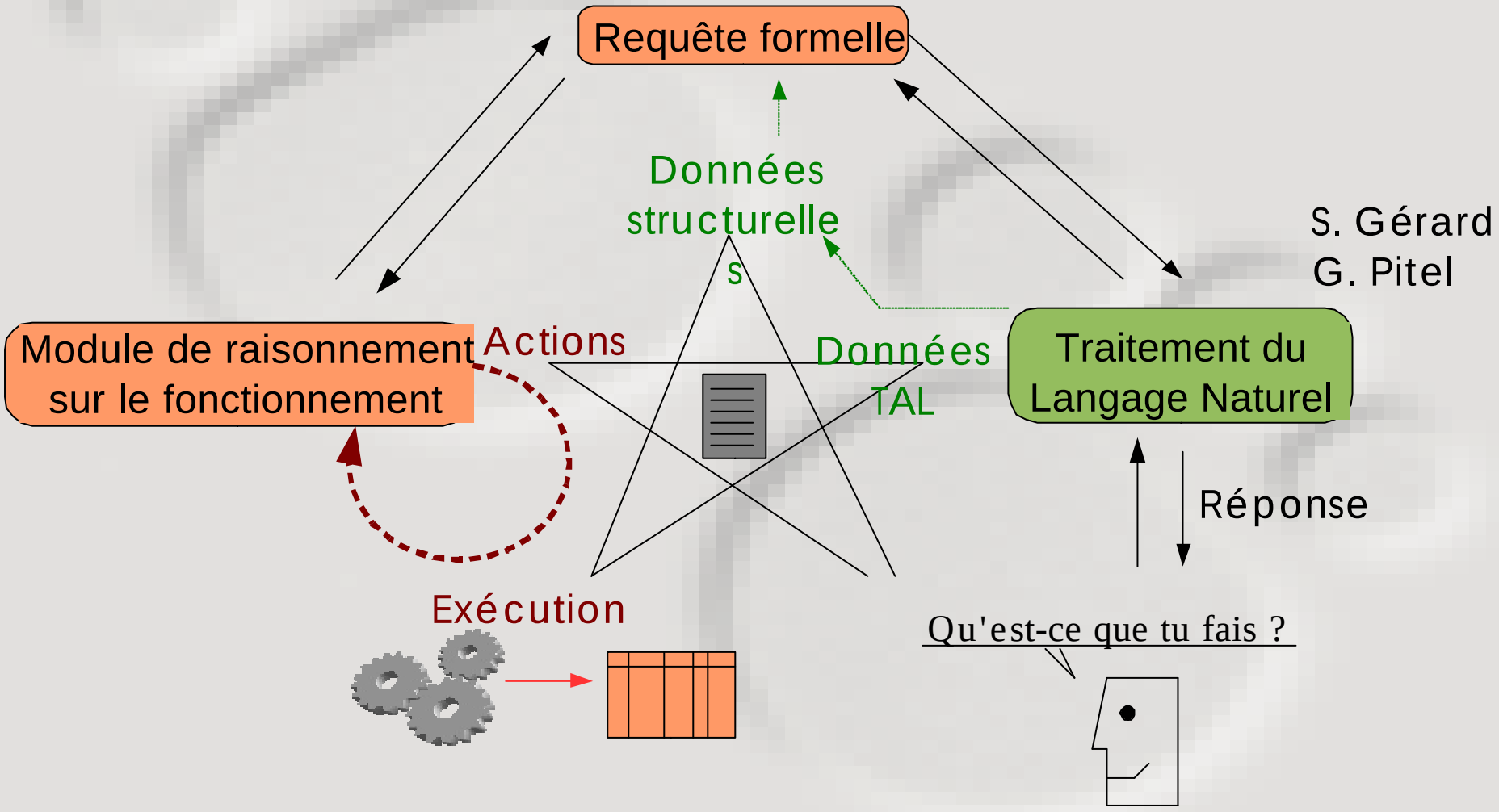
Projet InterViews

Échec de la doc en ligne

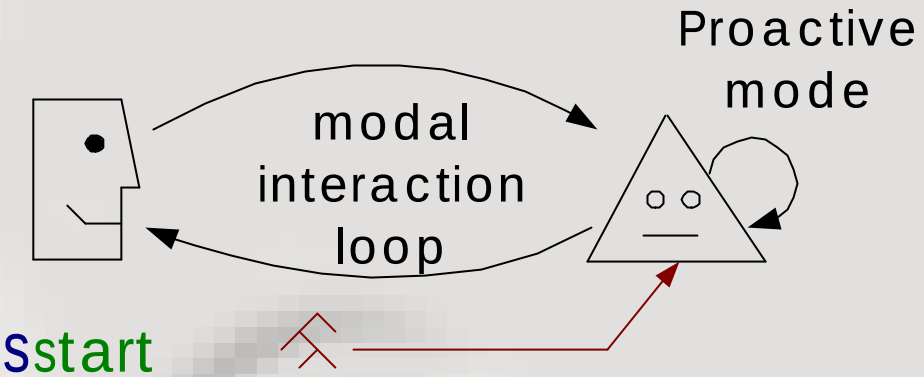
Agent dialogique

Composants actifs capables de répondre à des questions sur leur fonctionnement pour aider les utilisateurs ordinaires

Le projet *InterViews*



Modes d'interaction



- Préconditions
  - Événements externes
  - Gardes booléennes

cook

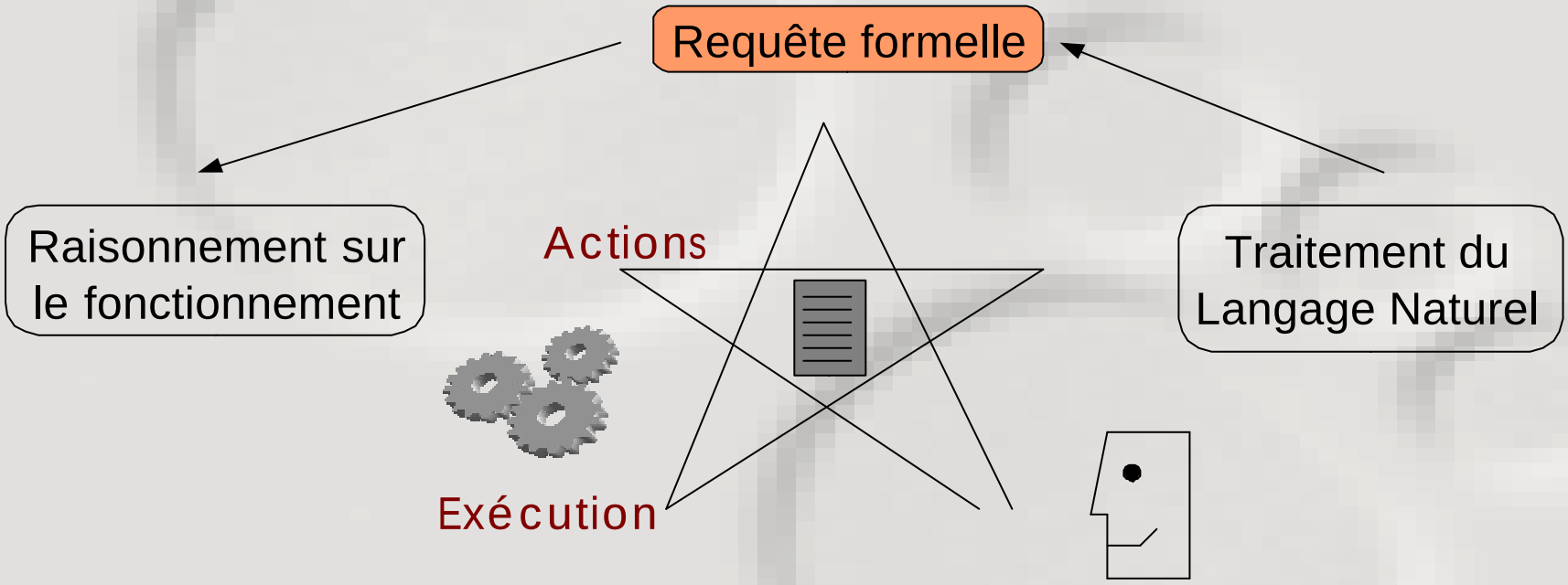
- Perception :
  - Comportement
  - Activité
  - État

Le four permet de cuire

La minuterie compte

La température est 170°C

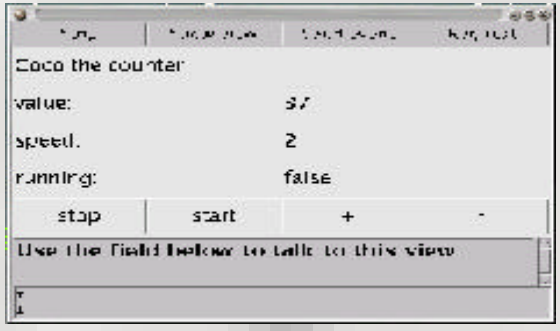
Un modèle de requêtes



Un modèle de requêtes (2)

Un premier corpus:

- What is your value ?
- What are you doing ?
- Can you count ?
- How can I stop you ?
- Why did you stop ?
- When will your value be 34 ?
- *etc.*



Classification:

- Acte de langage
- Fonctionnement
- Mécanismes de réponse

Langage de requêtes formelles

? = acte (performatif) ask, why...
? = type procédural état, activité...
? = sujet (ref. à un noeud)
? = objet (ens. de termes)
? = date passé, futur, relatif...
? = questions négatives

marqueurs

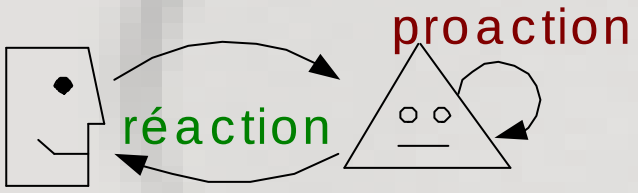
Principe général

- Acte → traitement effectué
  - Réponses directes:

    What = recherche d'éléments, **Ask** = avec filtre
  - Réponses indirectes:
    - Fondées sur la recherche d'éléments
    - **How** = plan (O-Plan), **Why** = diagnostic, explication

Shanahan, McIlraith

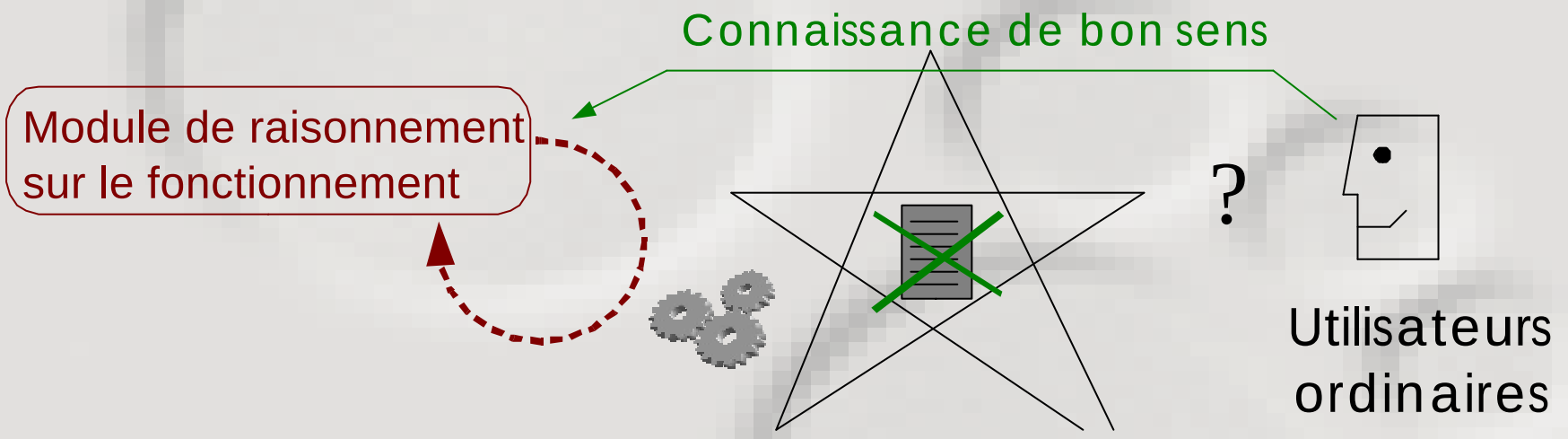
- Type → que chercher (réactions, processus...)



- Construction d'une réponse pertinente

  Toutes les infos nécessaires mais seulement cela

Requêtes de bon sens



Connaissances de bon sens

- Traitement de la référence
  - Données structurelles — the red cube
  - Actions « nommées » — count

  → Requêtes bien formées utilisant les K internes

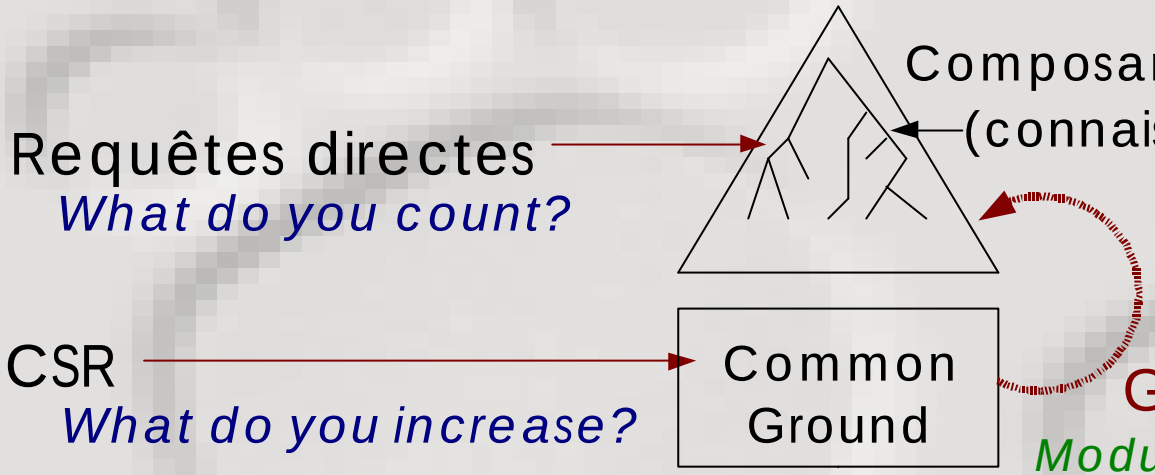
- Connaissances de bon sens
  - Non définie explicitement dans le composant
  - Compréhensible par la pluspart des humains

    Ex: *increase the value*, *count backwards*...

  → Requêtes formelles de bon sens (CSR) !

```
<Why,do,{view},{nothing},T,.>
<Ask,can,{view},{count[backwards]},T,.>
...
```

Notion de *Grinder*



Traitement des requêtes formelles

Grinder

Module d'interprétation sémantique

Interprète dans le contexte du composant en utilisant la sémantique opérationnelle

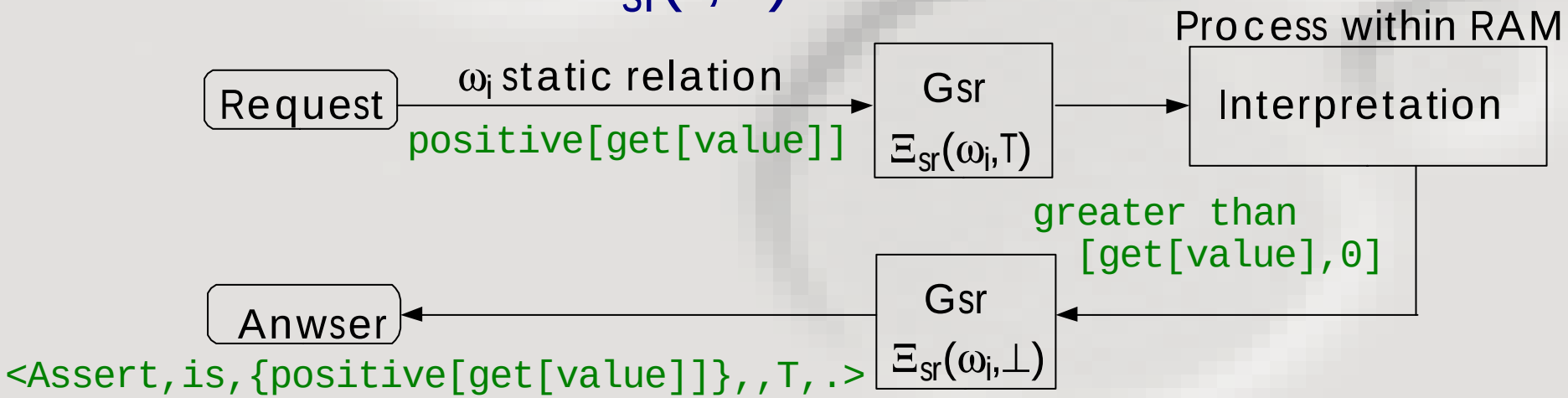
Corpus → trois grinders:

- Relations statiques — *Is your value positive?*
- Actions — *Did your value increase?*
- Relations procédurales — *Is start the inverse of stop?* *Do you do nothing?*

Relations statiques

Is your value *positive*?

- Gsr = ensemble de couples (s,r)
  - s = pattern de ground `positive[#1]`
  - t = pattern VDL bien défini `"greater than"[#1,0]`
- Conversion: $\Xi_{sr}(t,b)$

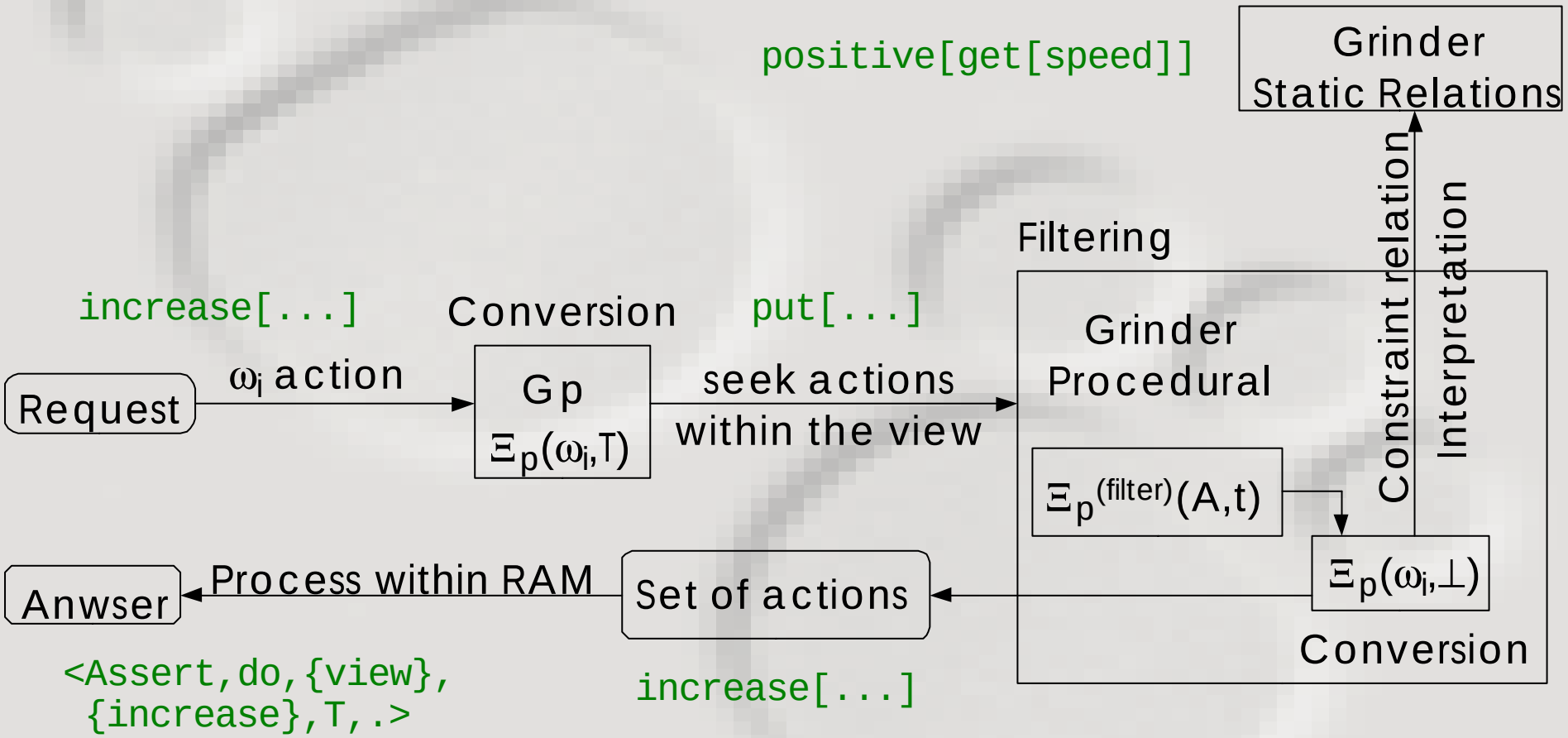


Le *grinder* des actions

Did your *increase* your value?

- Gp = ensemble de triplets (s,t,r)
  - s = pattern de ground `increase[path[#1],#2]`
  - t = pattern VDL
    `put[path[#1],plus[get[#1],#2]]`
  - r = relation de contrainte `positive[#2]`
- Termes non instanciés
  - *What do you increase?* `#1, #2`
  - *Did you increase your value?* `#2`

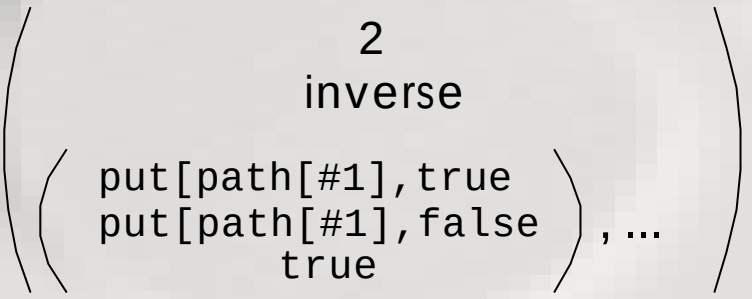
Algorithme


positive[get[speed]]
Grinder
Static Relations
Constraint relation
Interpretation
Filtering
increase[...]
Conversion
put[...]
Grinder
Procedural
Request
ωi action
Gp
Ξp(ωi,T)
seek actions
within the view
Ξp(filter)(A,t)
Ξp(ωi,⊥)
Conversion
Anwser
Process within RAM
Set of actions
<Assert,do,{view},
{increase},T,.>
increase[...]

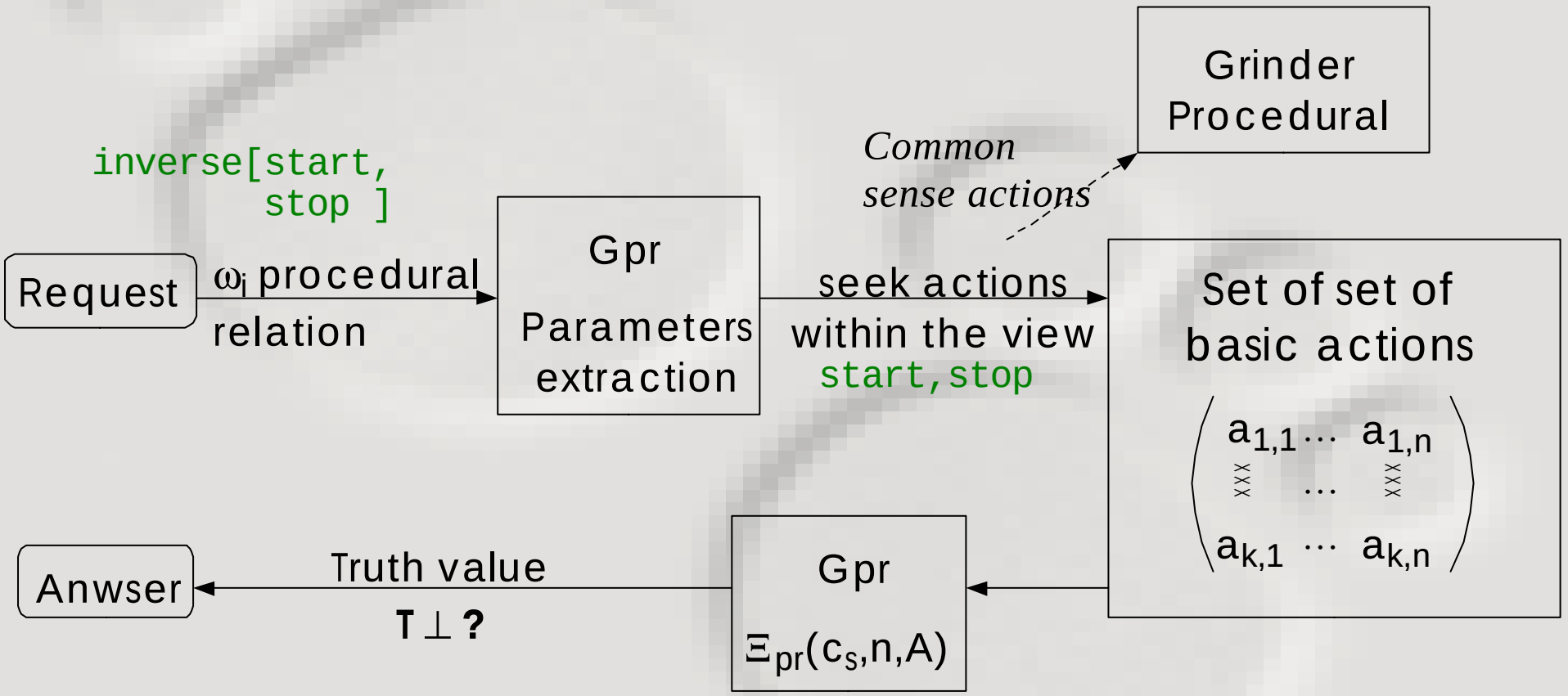
Relations procédurales

Is `start` the *inverse* of `stop`?

- Gpr = ensemble de triplets (n, c_s, R)
  - n = arité de la relation
  - cs = concept
  - R = ensemble de couples (T,r)
    - T = {t1,...,tn} = ensemble d'actions en relation
    - r = contrainte



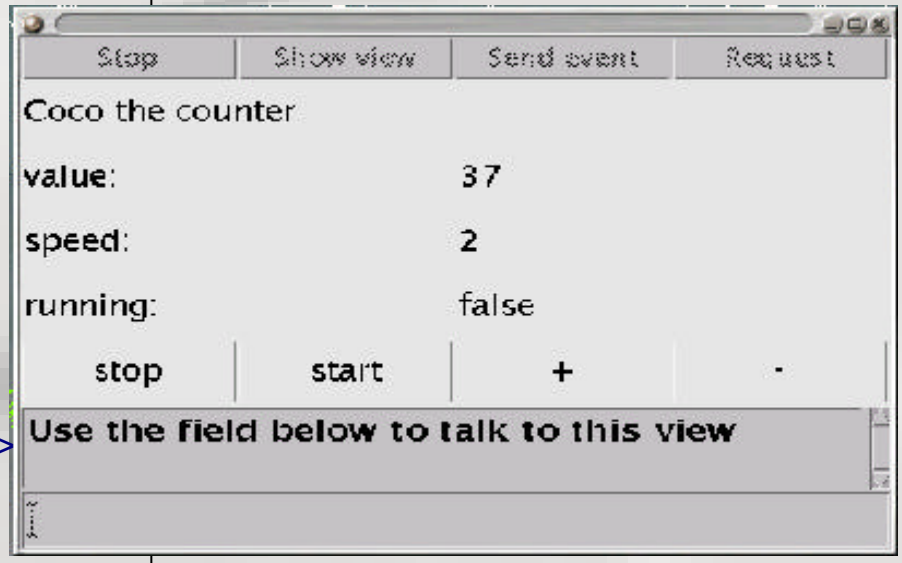
Algorithme


Grinder
Procedural
Common
sense actions
inverse[start,
stop]
Request
ωi procedural
relation
Gpr
Parameters
extraction
seek actions
within the view
start,stop
Set of set of
basic actions
a1,1 … a1,n
ak,1 … ak,n
Anwser
Truth value
T ⊥ ?
Gpr
Ξpr(cs,n,A)

Exemple

```
<view xmlns:vdl="http://www.limsi.fr/Individu/
          nico/xml/vdl.xsd">
 <variable type="integer" name="value">37</variable>
 <variable type="integer" name="speed">2</variable>
 <variable type="boolean" name="running">false</variable>
 <action><name>count</name>
  <guard><get>//variable[@name="running"]</get></guard>
  <put>
   <path>//variable[@name="value"]</path>
   <plus>
    <get>//variable[@name="value"]</get>
    <get>//variable[@name="speed"]</get>
   </plus>
  </put>
 </action>
 <action><name>stop</name> ... </action>
 <action><name>start</name>
  <event><start/><go/></event>
  <put> <path>//variable[@name="running"]</path>
      <value>false</value> </put>
 </action>
 <action><name>speed up</name> ... </action>
 <action><name>slow down</name> ... </action>
</view>
```

Est-ce que la valeur augmente ?



http://www.limsi.fr/Individu/nico/exemples/coco.html

Traitement

Do you increase your value?

`<Ask,do,{view},{increase[path[value]]},T,.>`

- L'objet de la requête contient `increase`

- Conversion en terme « bien défini »

  `put[path[value],plus[get[value]]]` `#2 = ?`

- Recherche de l'action (χ_{proc})

  Récupérer uniquement les *processus actifs*

  `count[put[path[value] ...]`

- Filtrage:
  - Conversion inverse en terme de « ground »

    `count[increase[path[value],get[speed]]]`
  - Vérification de la relation statique

    `positive[get[speed]]`

→ Réponse `<Assert,do,{view},{increase[path[value]]},T,.>`

Conclusion & perspectives

- Expressivité par rapport à notre corpus
  - Notions de bon sens dans questions & réponses
- Une première solution dans notre étude
  - Intégration dans les algos de réponse
- Un problème difficile
  - Vers une approche plus générique
    - Intégration avec les outils de TAL
    - Ontologies
  - Vers un grinder « dynamique » $\exists t_0 \text{ tq } \forall t>t_0,\ x_t \geq x_{t-1}$
  - Vers un grinder « évolutif », par apprentissage au cours de l'interaction