Generación de lenguaje natural

Kristina Striegnitz

Dept. of Computational Linguistics

Saarland University

kris@coli.uni-sb.de

http://www.coli.uni-sb.de/~kris

Qué es?

Generación de lenguaje natural es el área de lingüística computacional que trata de desarrollar métodos para automáticamente producir textos que expresan cierta información dada en una representación non-lingüística.

Estructura de la presentación

- Ejemplos de uso
- Una text adventure usando métodos de lingüística computacional
- De fórmulas lógicas a texto
- Posibles arquitecturas de sistemas de generación

Ejemplos de uso

Ejemplos de uso (1)

- FOG (Reiter, U. of Aberdeen, Escocia): pronósticos del tiempo para pescadores.
- input: datos numéricos sobre la temperatura, el viento, etc.
- output: textos cortos sobre el tiempo para una región específica
- puntos interesantes: El sistema tiene que extraer los datos importantes, resumirlos, y transformar datos numéricos en expresiones en lenguaje natural.

Ejemplos de uso (2)

- Presentación de demostraciones de teoremas lógicos en lenguaje natural (e.g. Fiedler, Horacek, Saarland U., Alemania).
- input: demostraciones de teoremas lógicos como las produce un demostrador automático de teoremas;
- output: una explicación de la demostración en lenguaje natural (adaptada para usuarios con diferentes niveles de experiencia)
- puntos interesantes: El sistema tiene que resumir los pasos de la demostración para que puedan ser expresados de una manera natural. Tiene que adaptar las palabras que usa y la especificidad de la explicación al nivel del usuario.

Ejemplos de uso (3)

- ILEX (Mellish, Oberlander, U. of Edinburgh, Escocia): descripciones de objetos en un museo.
- input: hechos sobre un objeto representado en una estructura de predicado y argumentos
- output: una descripción del objeto; puede referirse a descripciones de otros objetos que fueron generadas antes
- puntos interesantes: El sistema tiene que ordenar el hechos para que la descripción sea fluida.

Ejemplos de uso (4)

- TRIPS (Allen, Rochester U., EE.UU.): interfaz en lenguaje natural para un sistema que ayuda en planificar la evacuación de un área en caso de una catástrofe.
- input: información factual sobre la situación e información sobre el estado del diálogo con el usuario
- output: contestaciones en un diálogo
- puntos interesantes: El sistema tiene que producir enunciaciones que no solo expresan información factual pero que también sirven para desarrollar el diálogo.

Cuándo no usar generación

- No todos los sistemas que tienen texto como output necesitan métodos de generación.
- Si no hay mucha variabilidad en los textos que el sistema tiene que producir es más facil usar textos pre-fabricados.
- Estos textos pueden contener espacios a completar con pedazos de texto pre-fabricados.

Una text adventure usando métodos de lingüística computacional

Text adventures (1)

- Juegos de computadora en que toda la interacción es vía lenguaje natural.

Observation Lounge

This is where the station staff and visitors come to relax. There are a lot of tables and chairs here, a large observation window, and a plush carpet. In the corner you can see an AstroCola dispenser. A tube leads up to the station's main corridor.

> **put my galakmid coin into the dispenser**

Click. The dispenser display now reads "Credit = 1.00".

> **push diet astrocola button**

You hear a rumbling noise in the dispenser, but nothing appears in the tray.

> **kick dispenser**

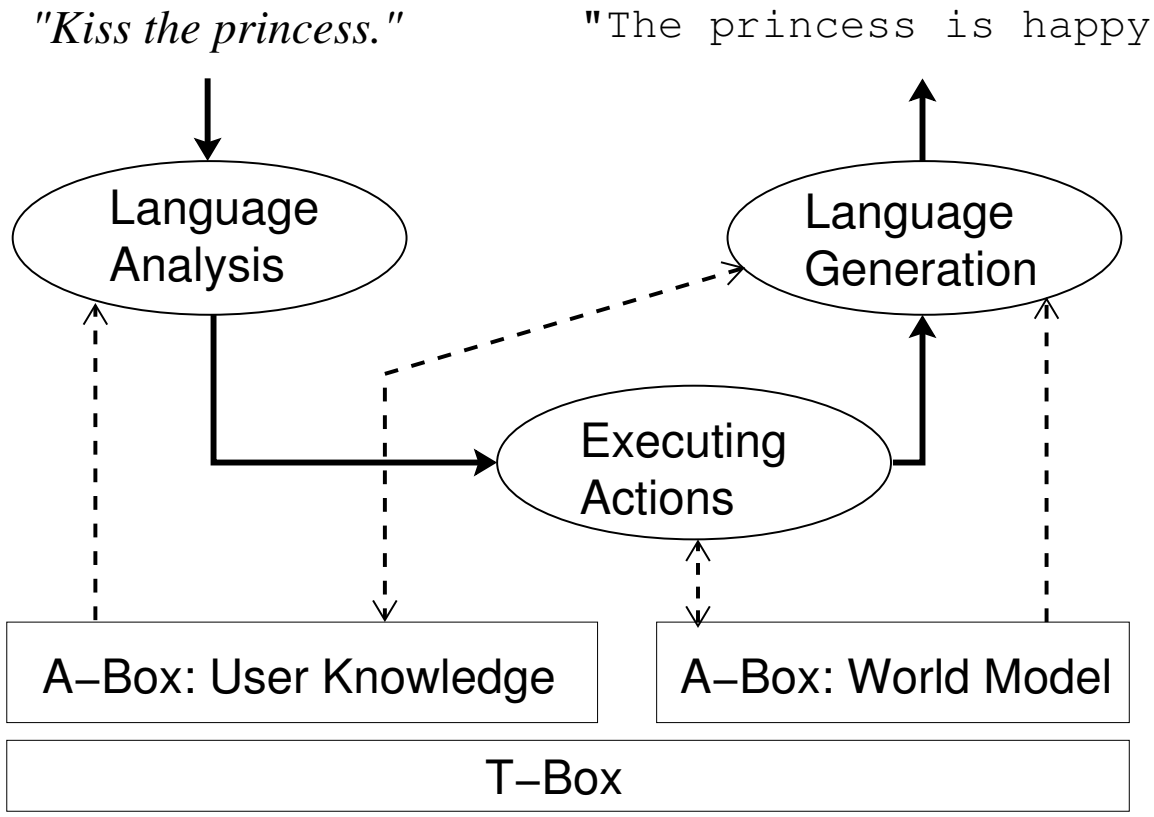
A can drops into the tray. Amazing! The oldest trick in the book, and it actually worked.

Text adventures (2)

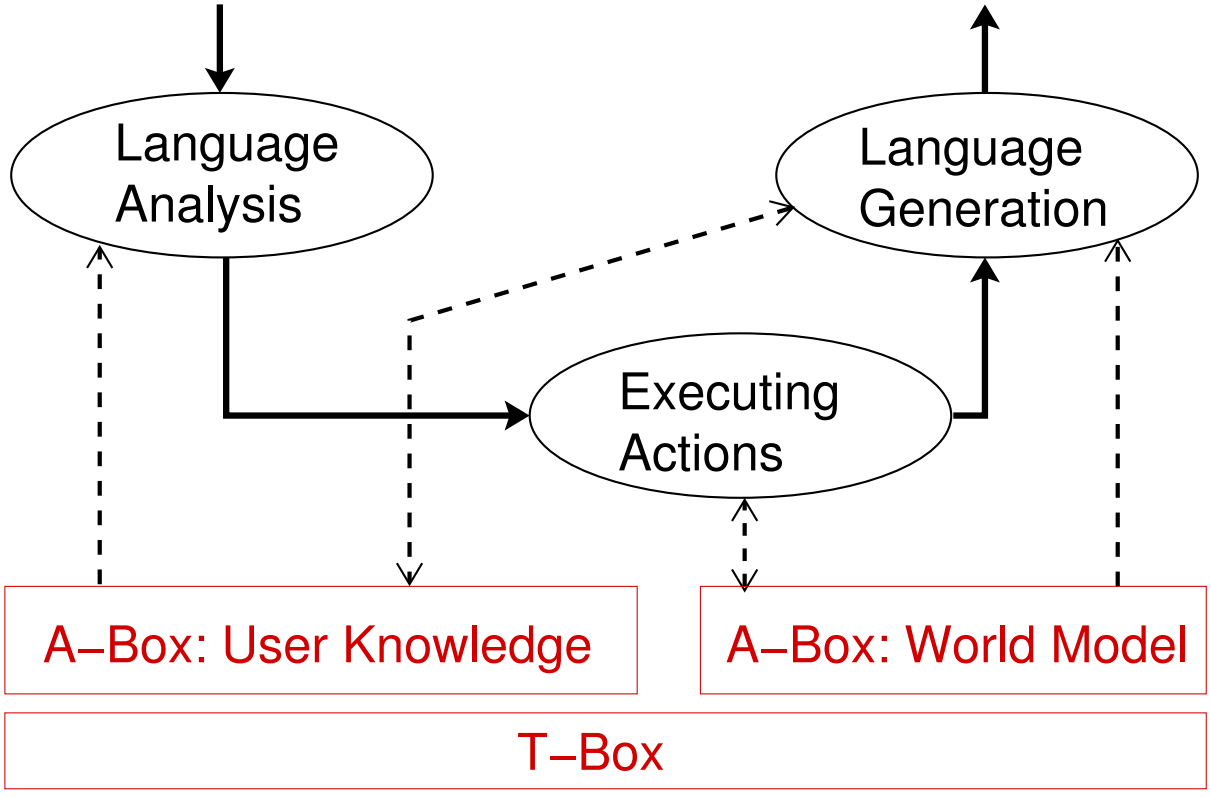
- Popular en los años 80 cuando las interfaces gráficas no eran posibles.
- No usaban métodos de lingüística computacional:
  - un parser muy básico para analizar el input
  - pedazos de texto pre-fabricados como output

Nuestra text adventure

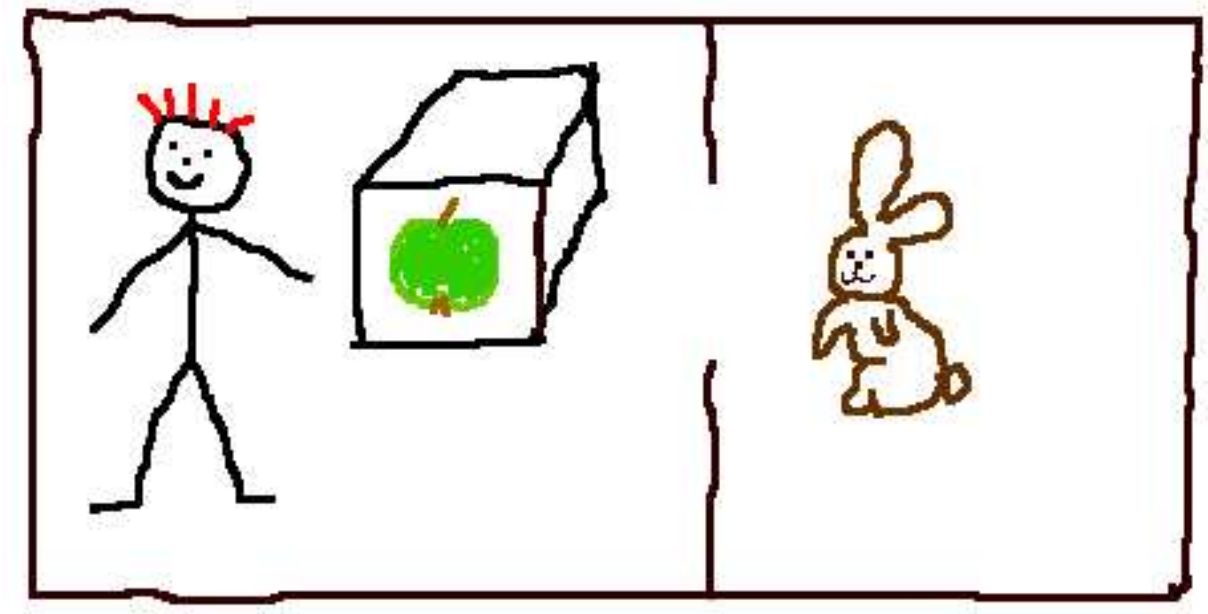
- Desarrollado por estudiantes de lingüística computacional en Saarbrücken.



Dónde estamos?



El A-Box representando el estado del mundo de juego



rabbit1:rabbit

box1:box

apple1:apple

apple1:green

myself:player

room1:room

room2:room2

(rabbit1,room2):has-location

(myself,room1):has-location

(box1,room1):has-location

(apple1,box1):has-location

El A-Box representando el conocimiento inicial del jugador

myself:player

La T-Box (1)

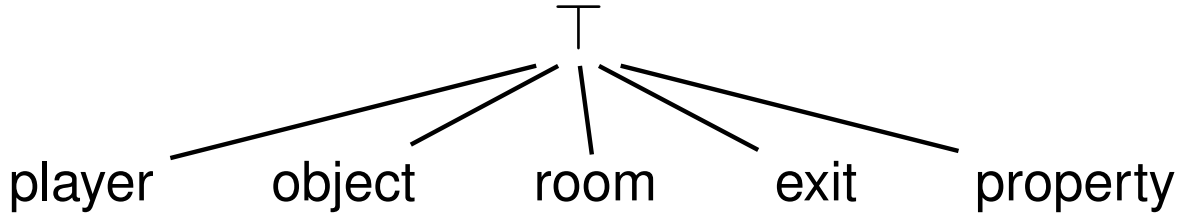
La T-Box define una jerarquía de conceptos para describir los objetos que aparecen en el juego.

$animal \sqsubseteq object$

$frog \sqsubseteq animal$

$alive \sqsubseteq property$

$\ldots$



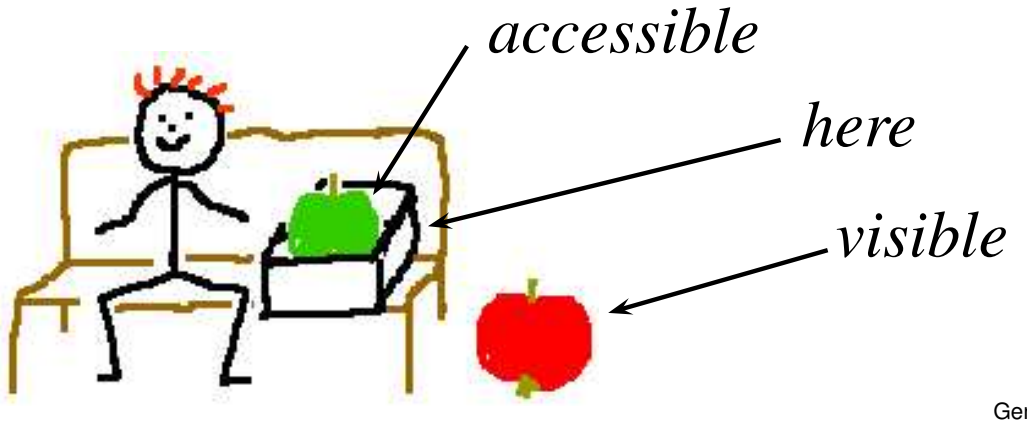
La T-Box (2)

También usamos la T-Box para definir algunos conceptos complejos que necesitamos para restringir las acciones del jugador.

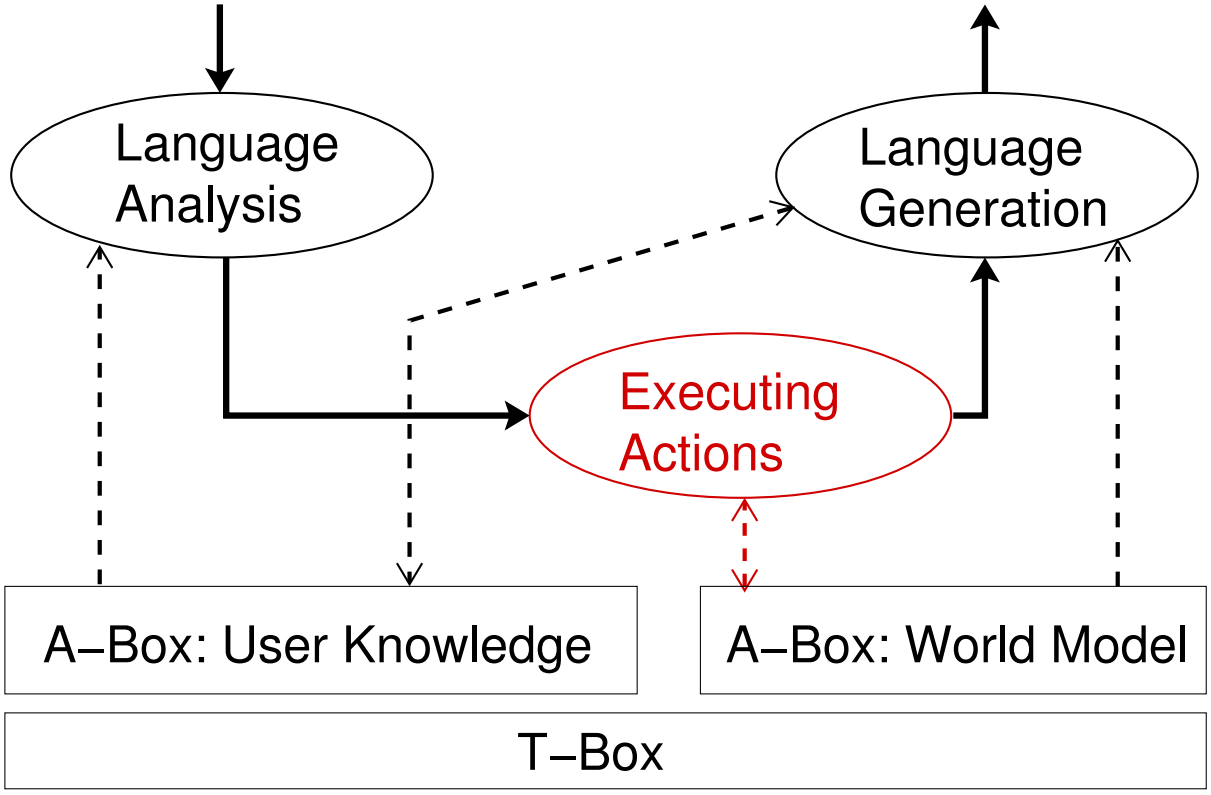
$$\mathit{here} \doteq \exists \mathit{has\text{-}location}^{-1}.\mathit{player}$$

$$\mathit{accessible} \doteq \mathit{here} \sqcup \exists \mathit{has\text{-}location}.\mathit{here} \sqcup \exists \mathit{has\text{-}location}.(\mathit{accessible} \sqcap \mathit{open})$$

$$\mathit{visible} \doteq \mathit{here} \sqcup \exists \mathit{has\text{-}location}.\mathit{here} \sqcup \exists \mathit{has\text{-}location}.(\mathit{visible} \sqcap \mathit{open}) \sqcup \exists \mathit{has\text{-}location}^{-1}.(\mathit{open} \sqcap \exists \mathit{has\text{-}location}^{-1}.\mathit{player}) \sqcup \exists \mathit{has\text{-}location}.\exists \mathit{has\text{-}location}^{-1}.(\mathit{open} \sqcap \exists \mathit{has\text{-}location}^{-1}.\mathit{player})$$

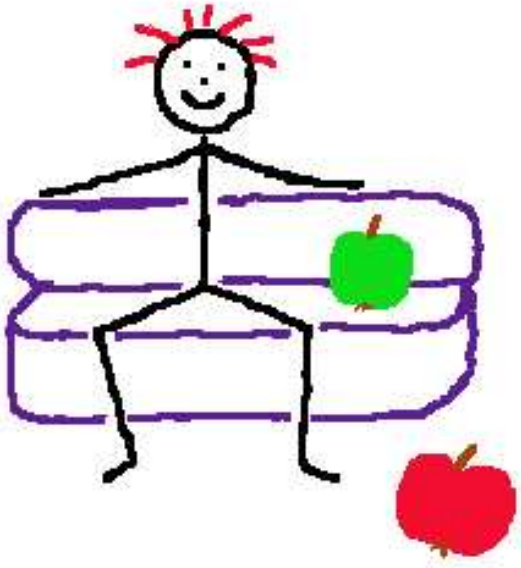


Dónde estamos?



Actuar en el mundo del juego

“take the green apple”



a:apple

a:green

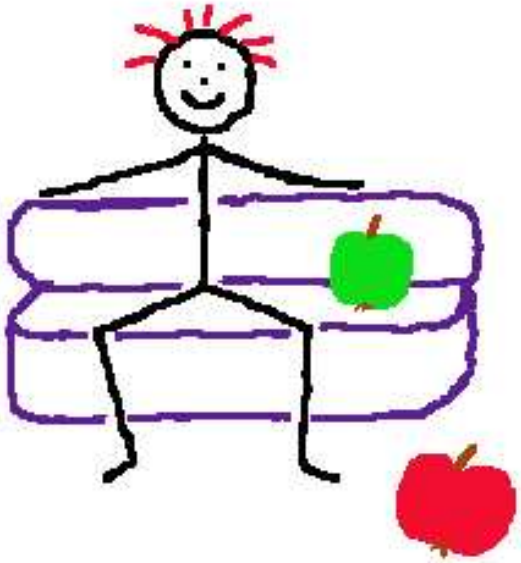
c:couch

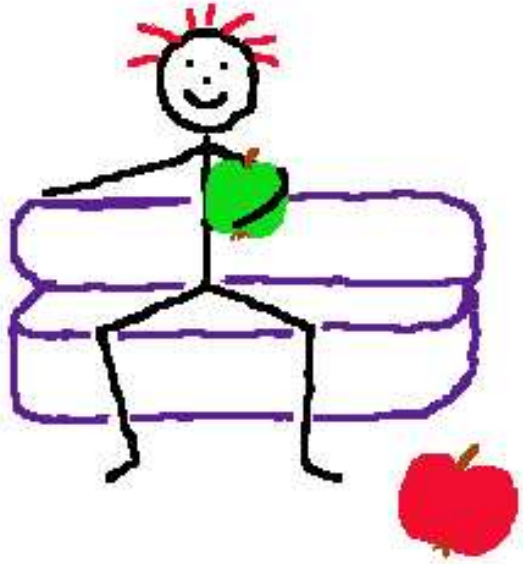
(a,c):has-location

. . .

Actuar en el mundo del juego

"take the green apple"





a:apple

a:green

c:couch

(a,c):has-location

. . .

a:apple

a:green

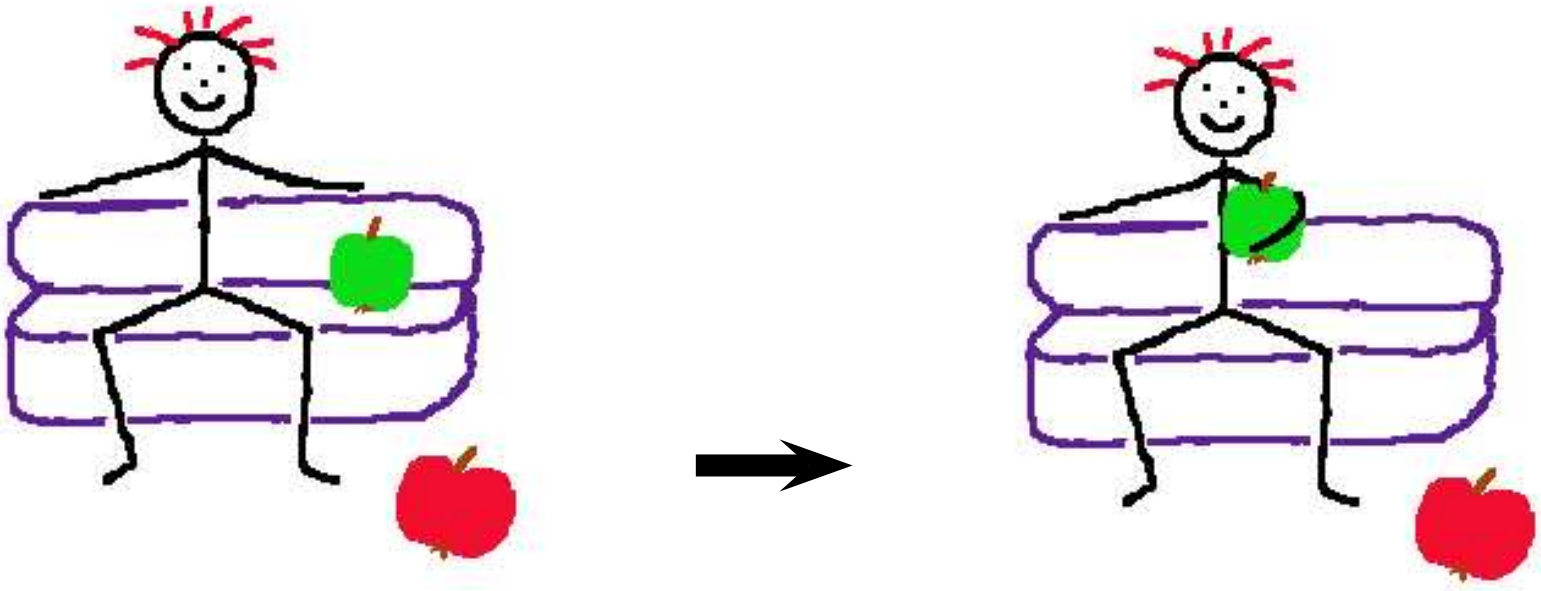
c:couch

(a,myself):has-location

. . .

Actuar en el mundo del juego

"take the green apple"



a:apple

a:green

c:couch

(a,c):has-location

...

take(X)
delete: *(X,Y):has-location*
add: *(X,myself):has-location*

a:apple

a:green

c:couch

(a,myself):has-location

...

Especificación de acciones

take(patient:X)	
preconditions:	*X:accessible, X:takeable, not(X:inventory-object)*
add-to-world:	*(X,myself):has-location*
delete-from-world:	*(X,Y):has-location*
add-to-user:	*(X,myself):has-location*
delete-from-user:	*(X,Y):has-location*

Especificación de acciones

"Take the apple."

análisis del input

take(patient:apple1)

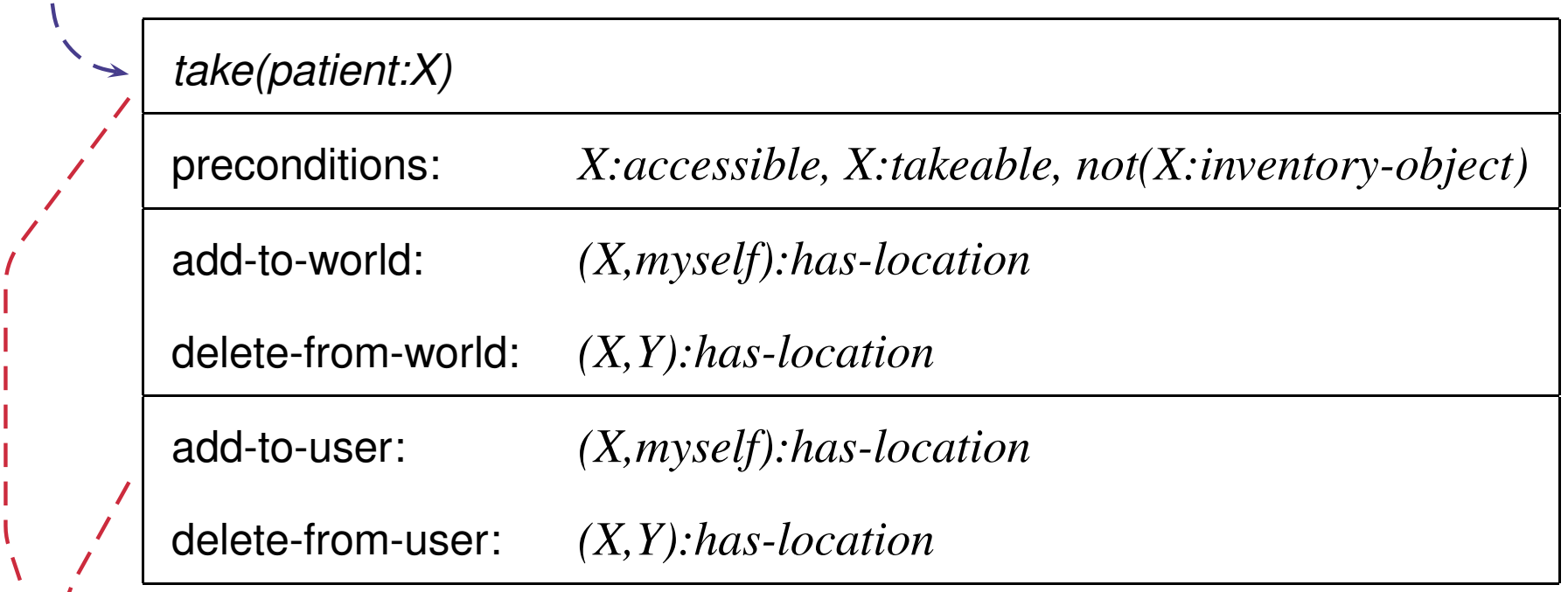
take(patient:X)	
preconditions:	*X:accessible, X:takeable, not(X:inventory-object)*
add-to-world:	*(X,myself):has-location*
delete-from-world:	*(X,Y):has-location*
add-to-user:	*(X,myself):has-location*
delete-from-user:	*(X,Y):has-location*

Especificación de acciones

"Take the apple."

análisis del input

take(patient:apple1)

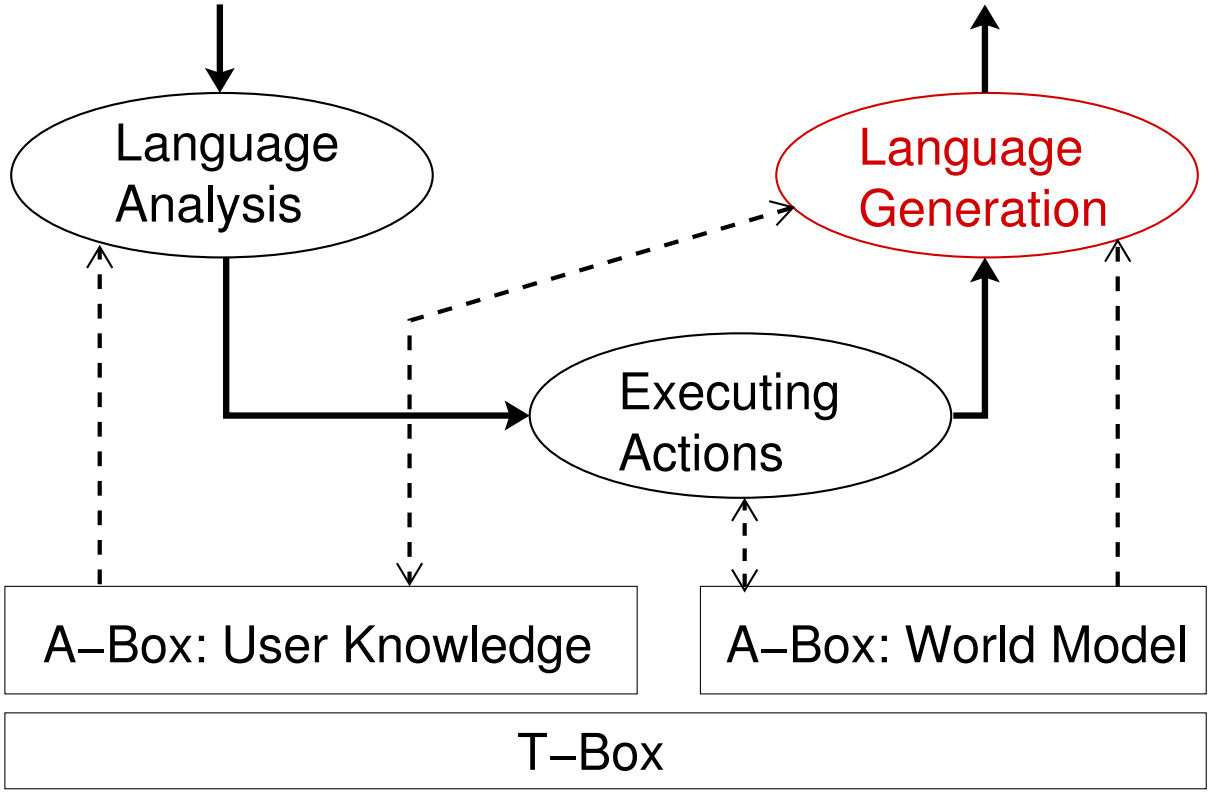


take(patient:X)	
preconditions:	*X:accessible, X:takeable, not(X:inventory-object)*
add-to-world:	*(X,myself):has-location*
delete-from-world:	*(X,Y):has-location*
add-to-user:	*(X,myself):has-location*
delete-from-user:	*(X,Y):has-location*

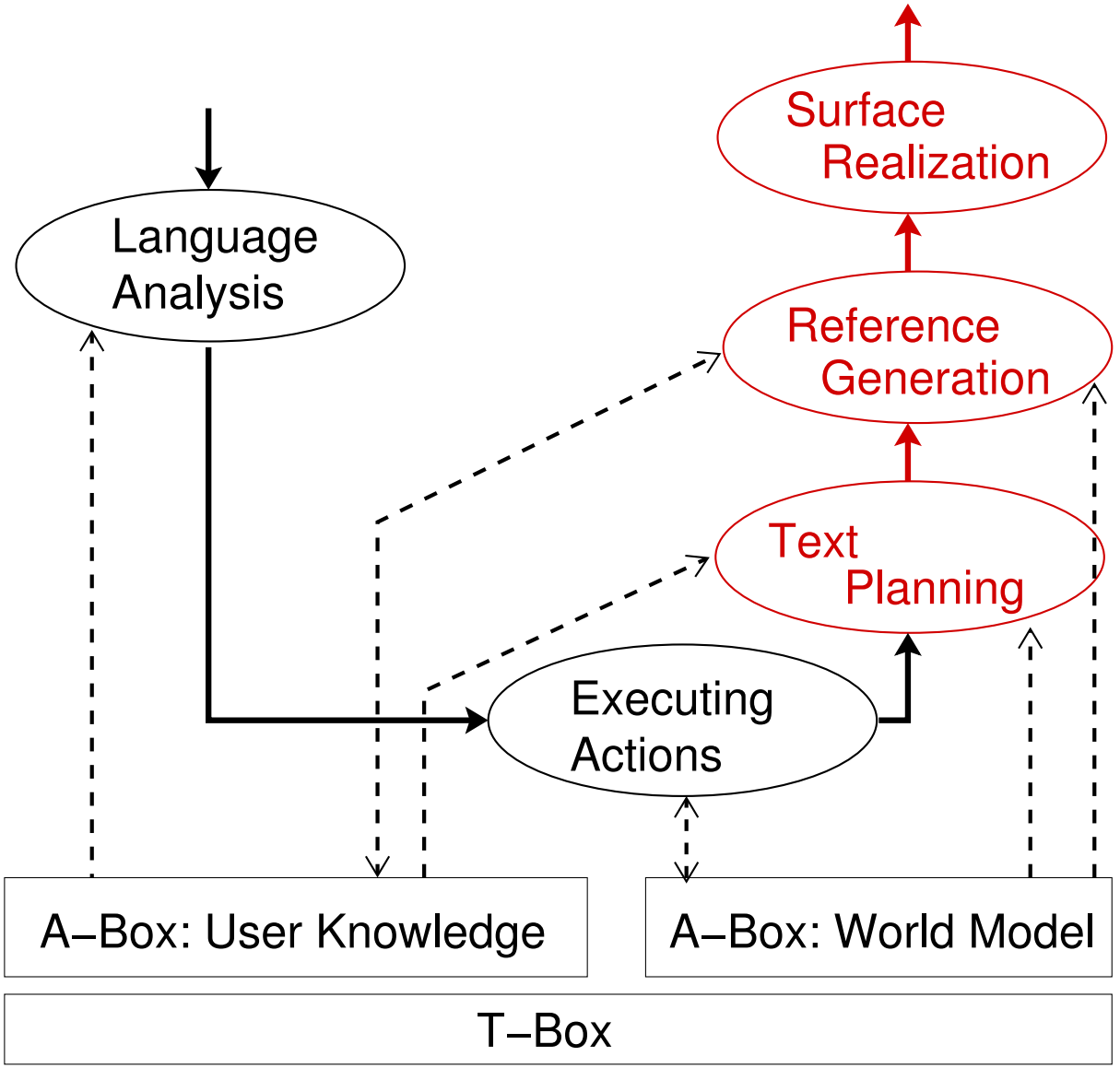
generación del output

De formulas lógicas a texto

Dónde estamos?



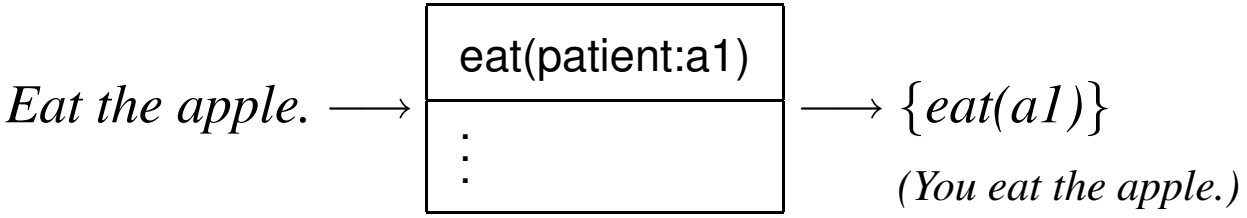
Generación en la text adventure



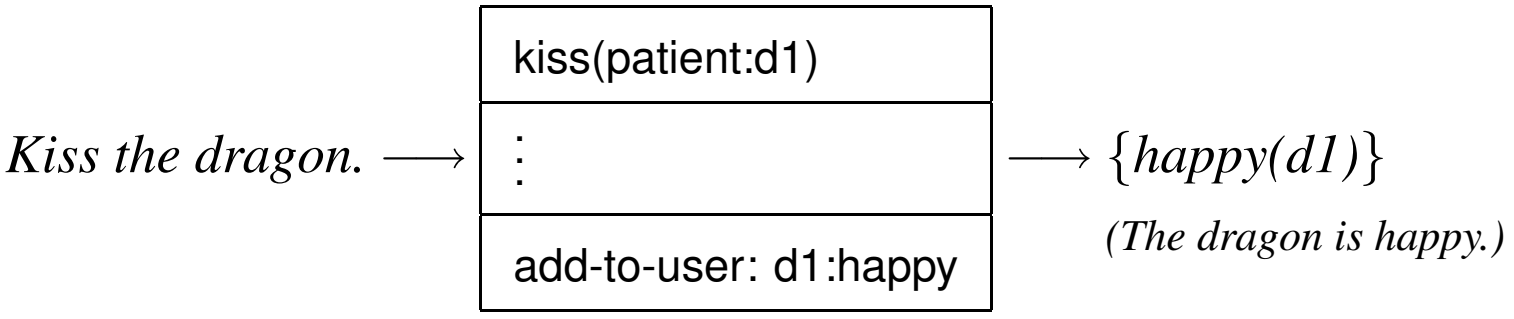
Selección del contenido (Content Determination)

- Determinar la información que hay que expresar.
- Depende del tipo de acción y de los cambios en el conocimiento del jugador.

Informar si la acción pudo realizarse:

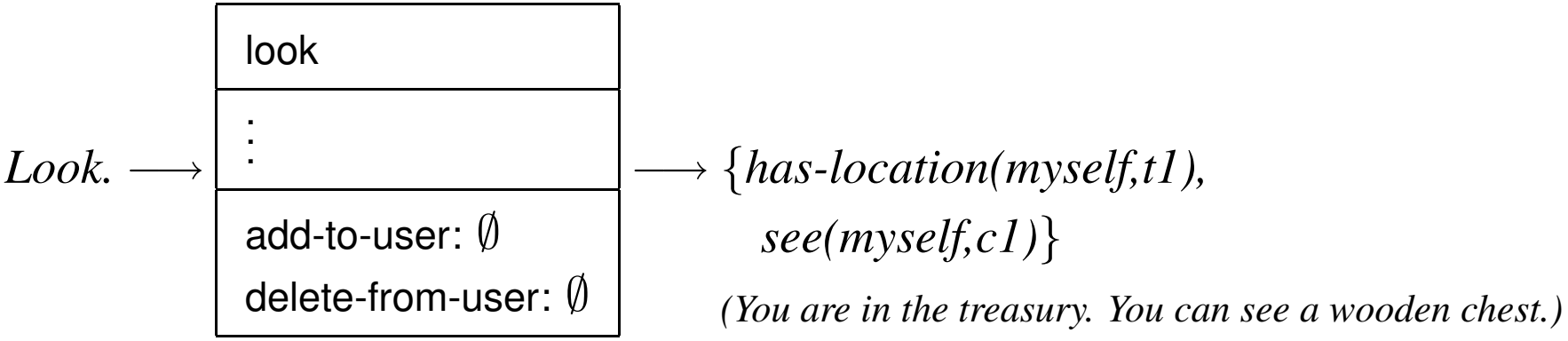


Describir cómo cambió el conocimiento del jugador:

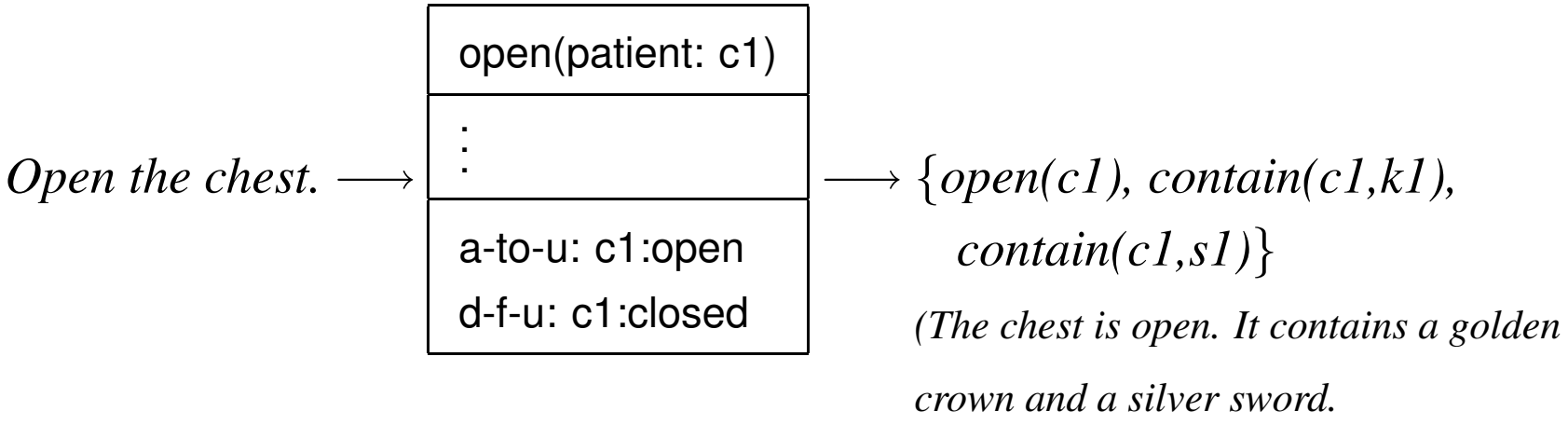


Selección del contenido (2)

Expresar un conjunto de información específico (describir el lugar actual del jugador, describir un objeto):



Combinaciones de estas opciones:



Organización del contenido (Document Structuring)

- Determinar el orden en que queremos presentar la información.

$Open\ the\ chest. \longrightarrow open(c1) \prec contain(c1,k1)$

(The chest is open. It contains a golden crown.)

es mejor que

$Open\ the\ chest. \longrightarrow contain(c1,k1) \prec open(c1)$

(The chest contains a golden crown. It is open.)

Organización del contenido (Document Structuring)

- Determinar el orden en que queremos presentar la información.

$$\textit{Open the chest.} \longrightarrow \textit{open(c1)} \prec \textit{contain(c1,k1)}$$

(The chest is open. It contains a golden crown.)

es mejor que

$$\textit{Open the chest.} \longrightarrow \textit{contain(c1,k1)} \prec \textit{open(c1)}$$

(The chest contains a golden crown. It is open.)

- Selección del contenido + Organización del contenido = Planificación del texto (Text Planning)
- En la text adventure tratamos la selección y la organización del contenido al mismo tiempo.

Planificación del texto por esquemas

- Un esquema es un modelo para un cierto tipo de texto.
- Elegimos esquemas dependiendo del tipo de acción efectuada.
- Tenemos que llenar los esquemas con la información que corresponde a la situación actual.
- Los esquemas más importantes usados en la text adventure:

ReportAction expresa que la acción pudo realizarse.

ReportUserAdd expresa todos los hechos que añadimos al A-Box del jugador.

DescribeLocation es un esquema complejo para describir el lugar de un objeto. Accede los esquemas 'DescribeContents' y (recursivamente) 'DescribeLocation'.

DescribeContents describe los objetos que están en un objeto dado.

DescribeObject es un esquema complejo para dar una descripción detallada de un objeto.

Ejemplo

DescribeLocation(O1)

output: $has\text{-}location(O1,Loc1)$

DescribeProperties(O1)

output: $\{P_1(O1), \ldots, P_n(O1)\}$

DescribeContents(Loc1)

output: $\{has\text{-}location(X,Loc1) \mid X \neq O1\}$

DescribeExits(Loc1)

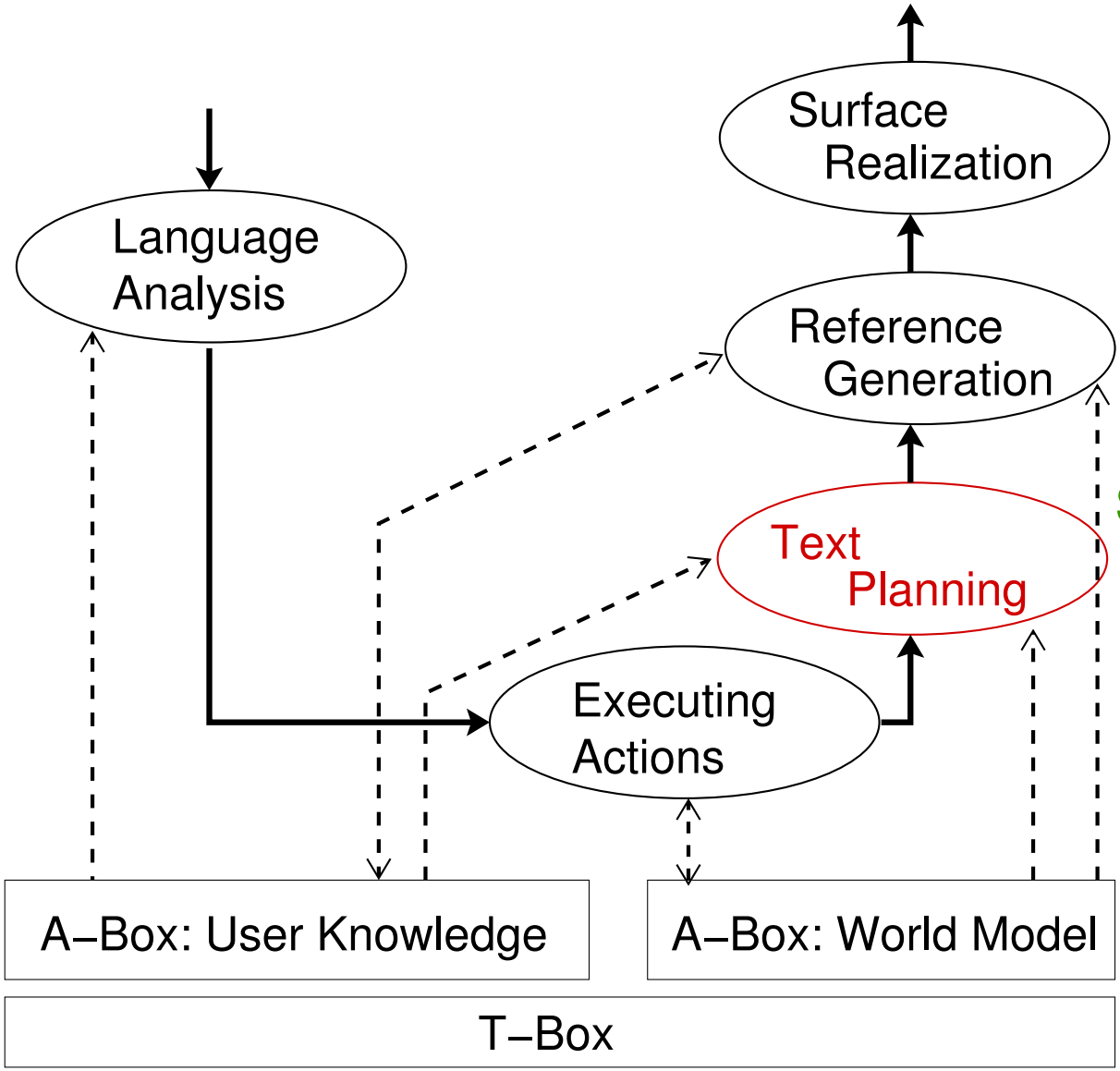
output: $\{ has\text{-}exit(Loc1,E_1), \ldots, has\text{-}exit(Loc1,E_m)\}$

DescribeLocation(Loc1)

output: descripción de Loc1, si Loc1 está abierto

si no está abierto, parar.

Dónde estamos?



Selección del contenido

Organización del contenido

Distribución del contenido en frases (Aggregation)

Open the chest. $\longrightarrow$ $open(c1) \prec contain(c1,\{k1,s1\})$

(The chest is open. It contains a golden crown and a silver sword.)

es mejor que

Open the chest. $\longrightarrow$ $open(c1) \prec contain(c1,k1) \prec contain(c1,s1)$

(The chest is open. It contains a golden crown. It contains a silver sword.)

Distribución del contenido en frases (2)

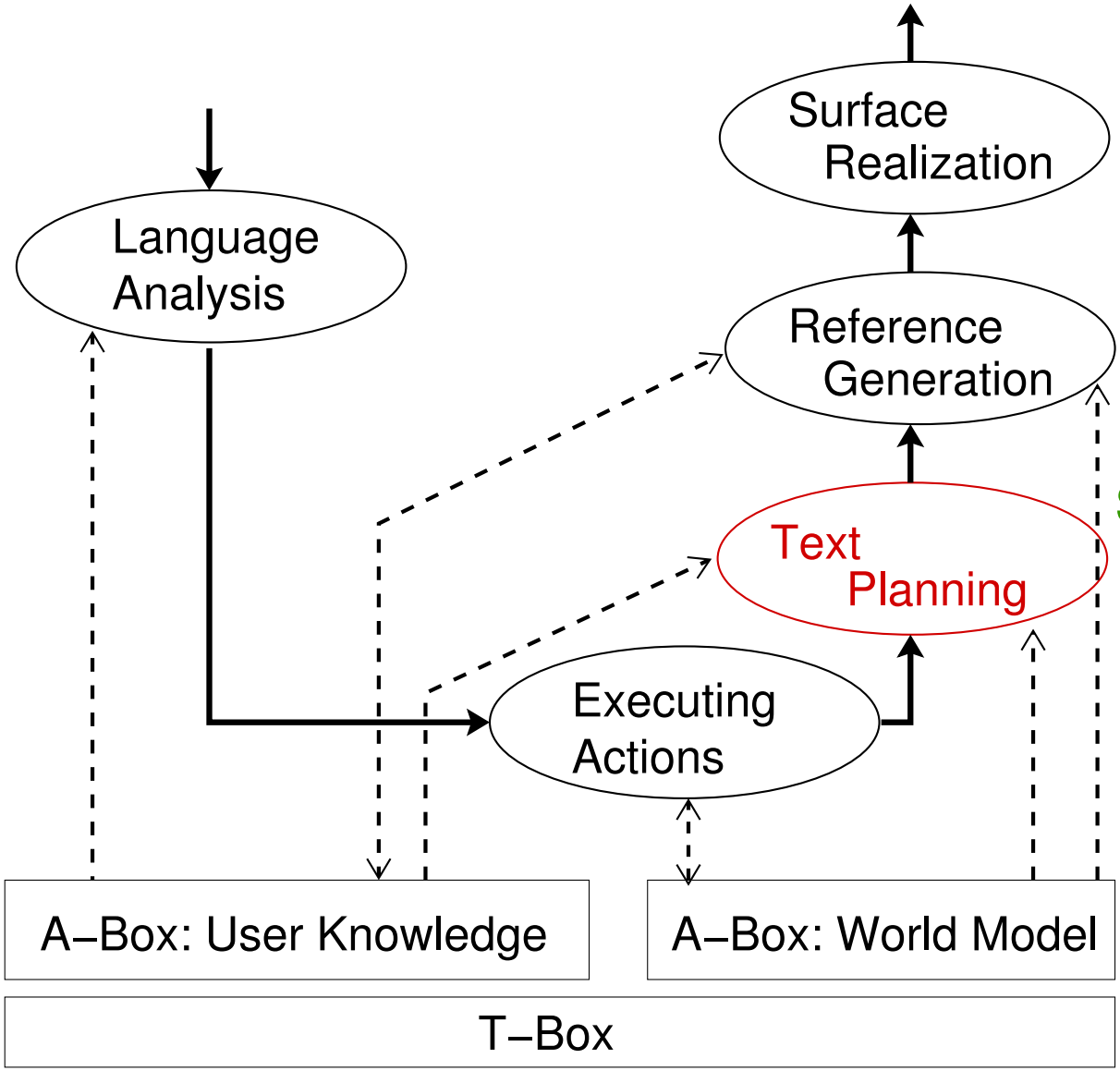
- En el juego usamos reglas del tipo:
  - Reunir objetos que tienen la misma relación con el objeto que estamos describiendo.

    Ej.: $contain(c1,k1) \prec contain(c1,s1) \Rightarrow contain(c1,\{k1,s1\})$
  - Si estamos describiendo un objeto, reunir propiedades de este objeto.

    Ej.: $ugly(w1) \prec alive(w1) \Rightarrow \{ugly(w1),\ alive(w1)\}$

    (The worm, which is ugly, is alive.)
- La etapa de agregación es muy importante para que el texto sea fluido.
- Todavía no se sabe bien cómo realizarla en el caso general.

Dónde estamos?



Selección del contenido

Organización del contenido

Distribución del contenido

Qué tenemos?

$$\textit{has-location(myself,t1)} \prec \textit{see(myself,\{c1,d1\})} \prec \textit{has-exit(t1,e1)}$$

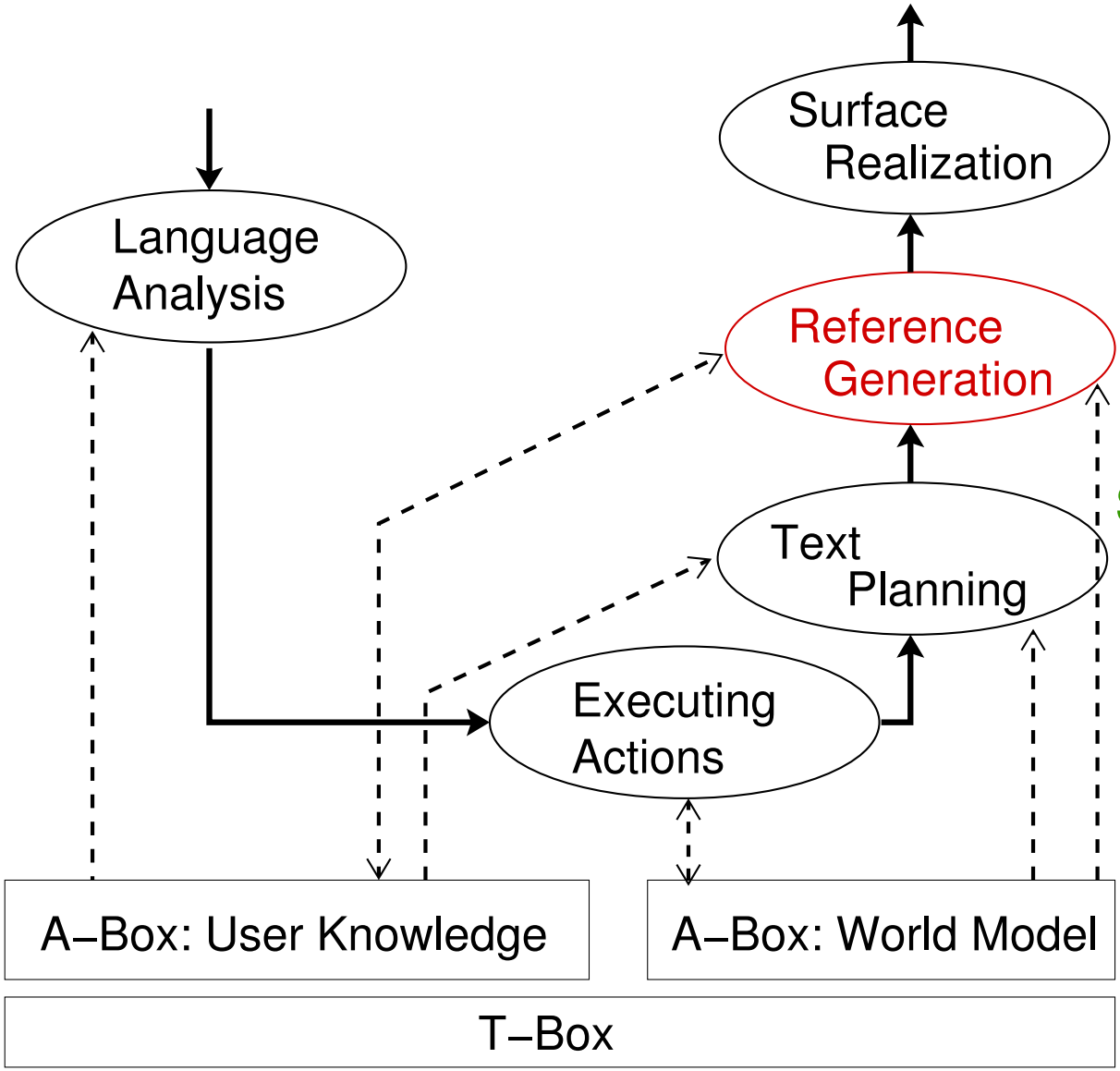
- Todavía no podemos producir frases completas:

  You are in [t1]. *You can see* [c1] *and* [d1]. [t1] *has an exit* [e1].

- t1, c1, d1, y e1 son nombres internos para individuos.

  Necesitamos expresiones nominales para referir a estos individuos.

Dónde estamos?



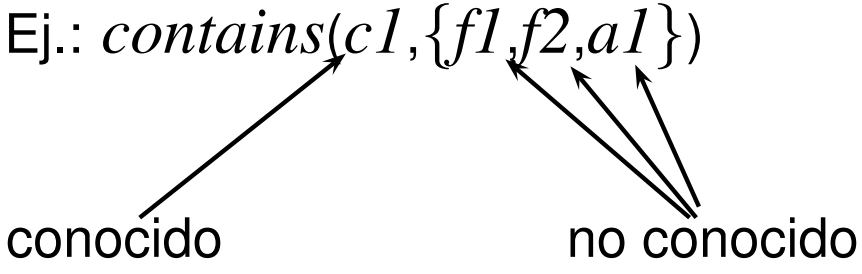
Selección del contenido

Organización del contenido

Distribución del contenido

Generación de expresiones referenciales
(Generation of referring expressions, GRE)

- Cómo referir a individuos que el jugador todavía no conoce?
- Cómo referir a individuos que el jugador ya conoce?



Introducir objetos

- Si el jugador todavía no conoce el objeto, usamos una expresión nominal indefinida.
- Decimos que tipo de objeto es y de que color es (si tiene color).

Ej.: $\{contains(c1,\{f1,f2,a1\})\}$

$+ \quad \{indef(f1), frog(f1), green(f1)\}$

$+ \quad \{indef(f2), frog(f2), brown(f2)\}$

$+ \quad \{indef(a1), apple(a1), red(a1)\}$

Referir a objetos ya introducidos

- Si el jugador ya conoce el objeto, usamos una expresión nominal definida.
- Tenemos que incluir suficiente información para que el objeto se distinga de todos los otros objetos que el jugador conoce.

Referir a objetos ya introducidos (2)

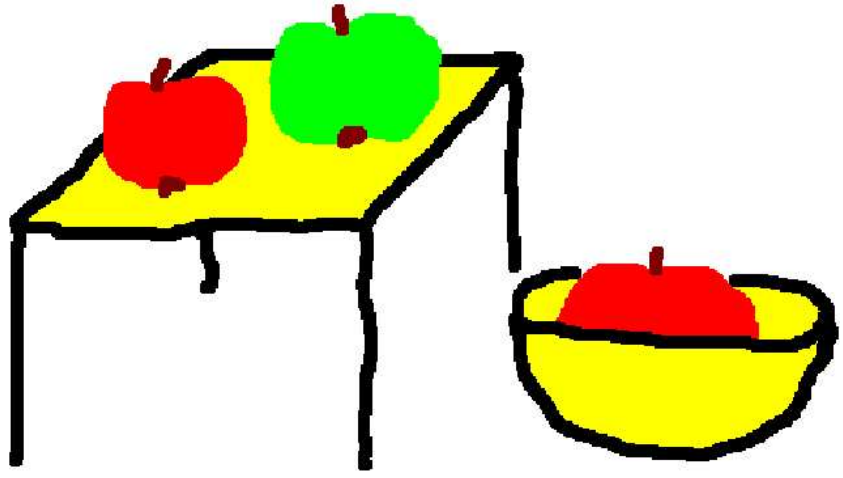
A-Box del jugador:

apple(a1), *green(a1)*, *has-location(a1,t1)*,

apple(a2), *red(a2)*, *has-location(a2,t1)*,

apple(a3), *red(a3)*, *has-location(a3,b1)*,

table(t1), *bowl(b1)*



Referir a objetos ya introducidos (2)

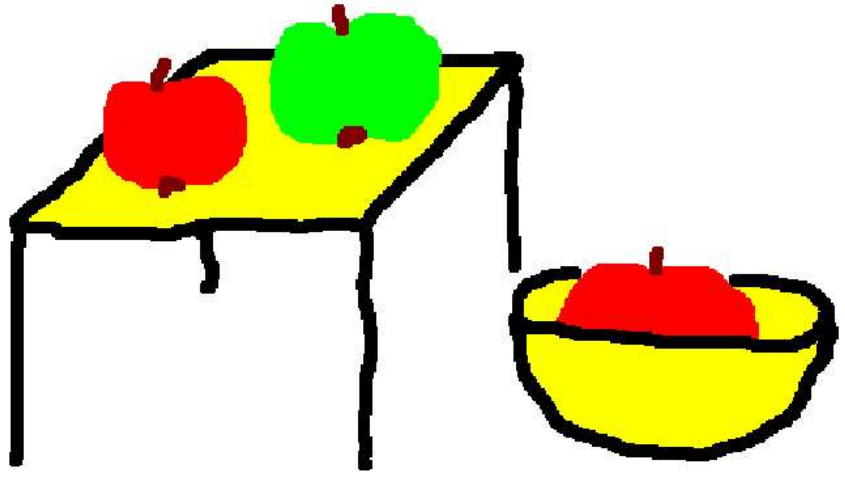
A-Box del jugador:

apple(a1), *green(a1)*, *has-location(a1,t1)*,

apple(a2), *red(a2)*, *has-location(a2,t1)*,

apple(a3), *red(a3)*, *has-location(a3,b1)*,

table(t1), *bowl(b1)*



para describir *a1*: *green(a1)* *(the green [thing???])*

Referir a objetos ya introducidos (2)

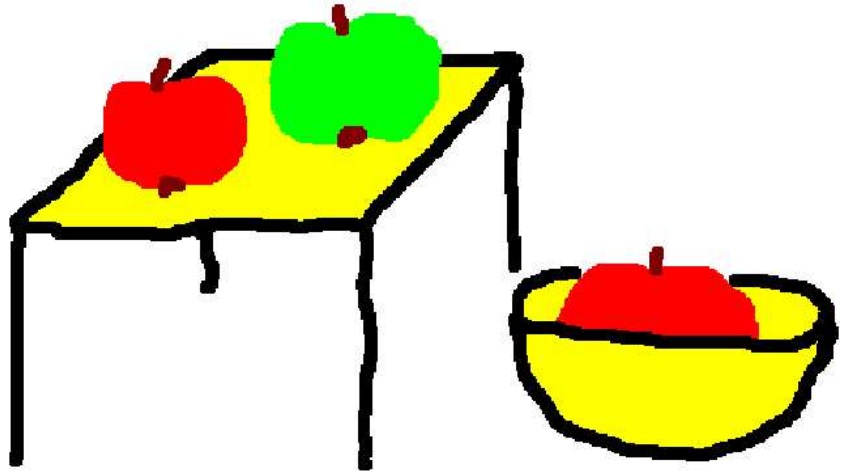
A-Box del jugador:

apple(a1), green(a1), has-location(a1,t1),

apple(a2), red(a2), has-location(a2,t1),

apple(a3), red(a3), has-location(a3,b1),

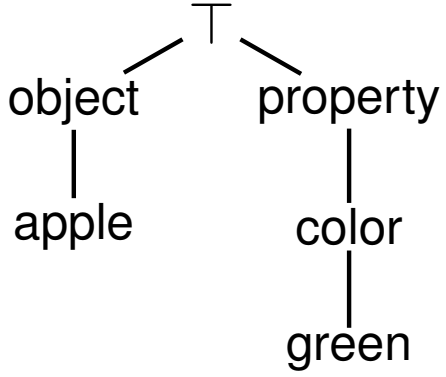
table(t1), bowl(b1)



para describir *a1*: *green(a1)* *(the green [thing???])*

o mejor: *apple(a1), green(a1)* *(the green apple)*

La T-Box nos deja distinguir entre tipos de objetos y otros propiedades:



Referir a objetos ya introducidos (2)

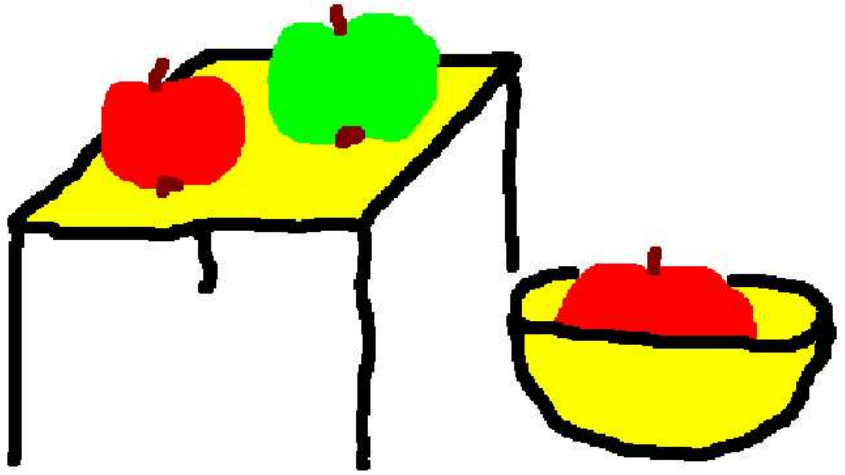
A-Box del jugador:

apple(a1), green(a1), has-location(a1,t1),

apple(a2), red(a2), has-location(a2,t1),

apple(a3), red(a3), has-location(a3,b1),

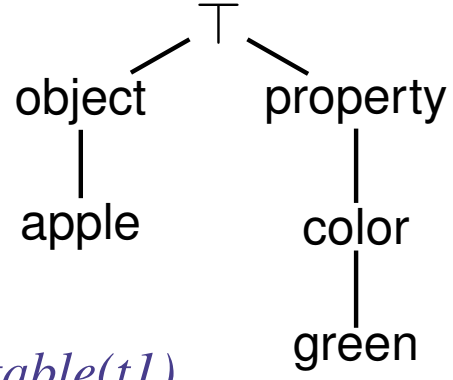
table(t1), bowl(b1)



para describir *a1*: *green(a1)* *(the green [thing???])*

o mejor: *apple(a1), green(a1)* *(the green apple)*

La T-Box nos deja distinguir entre tipos de objetos y otros propiedades:



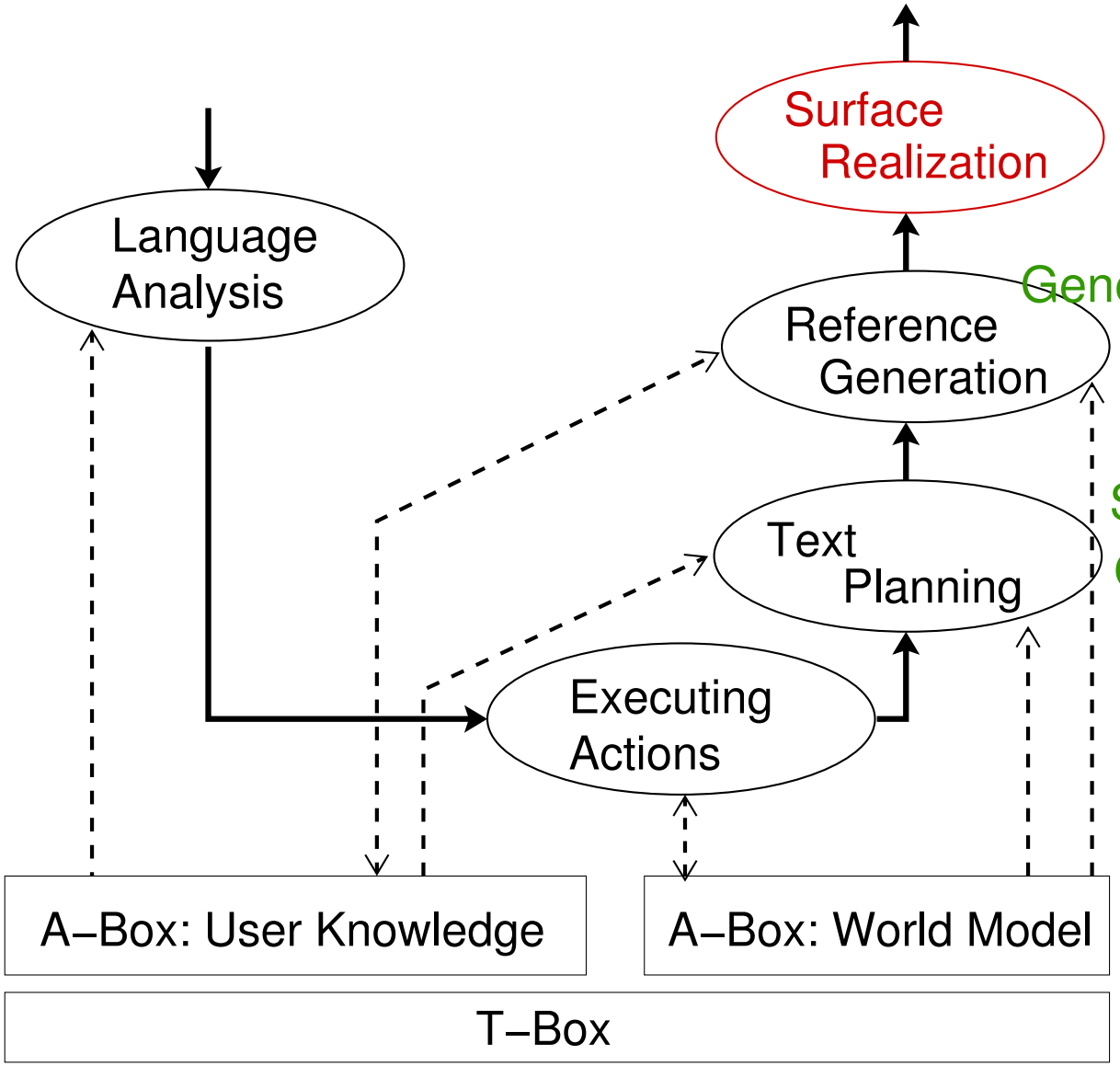
para describir *a2*: *apple(a2), red(a2), has-location(a2,t1), table(t1)*

(the red apple on the table)

Qué tenemos?

$$\{contains(c1,\{f1,f2,a1\}),$$
$$indef(f1), frog(f1), green(f1),$$
$$indef(f2), frog(f2), brown(f2),$$
$$indef(a1), apple(a1), red(a1),$$
$$def(c1), couch(c1)\}$$

Dónde estamos?



Generación de expresiones nomir

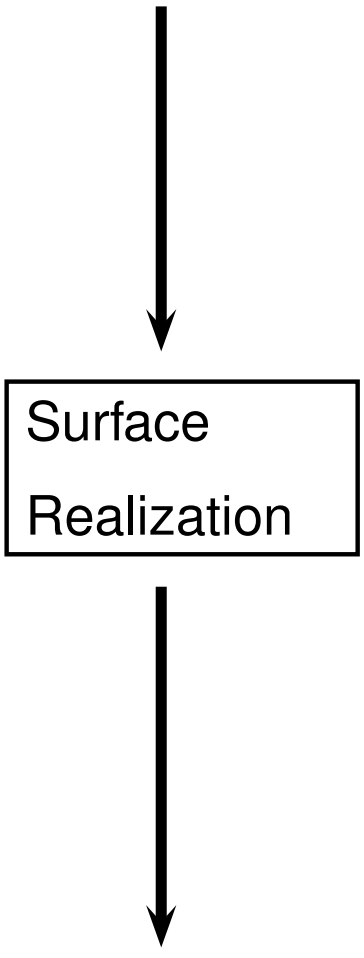
Selección del contenido

Organización del contenido

Distribución del contenido

Realización del texto (Surface Realization)

$\{contains(ch1,c1), def(ch1), chest(ch1), indef(c1), crown(c1)\}$



The chest contains a crown.

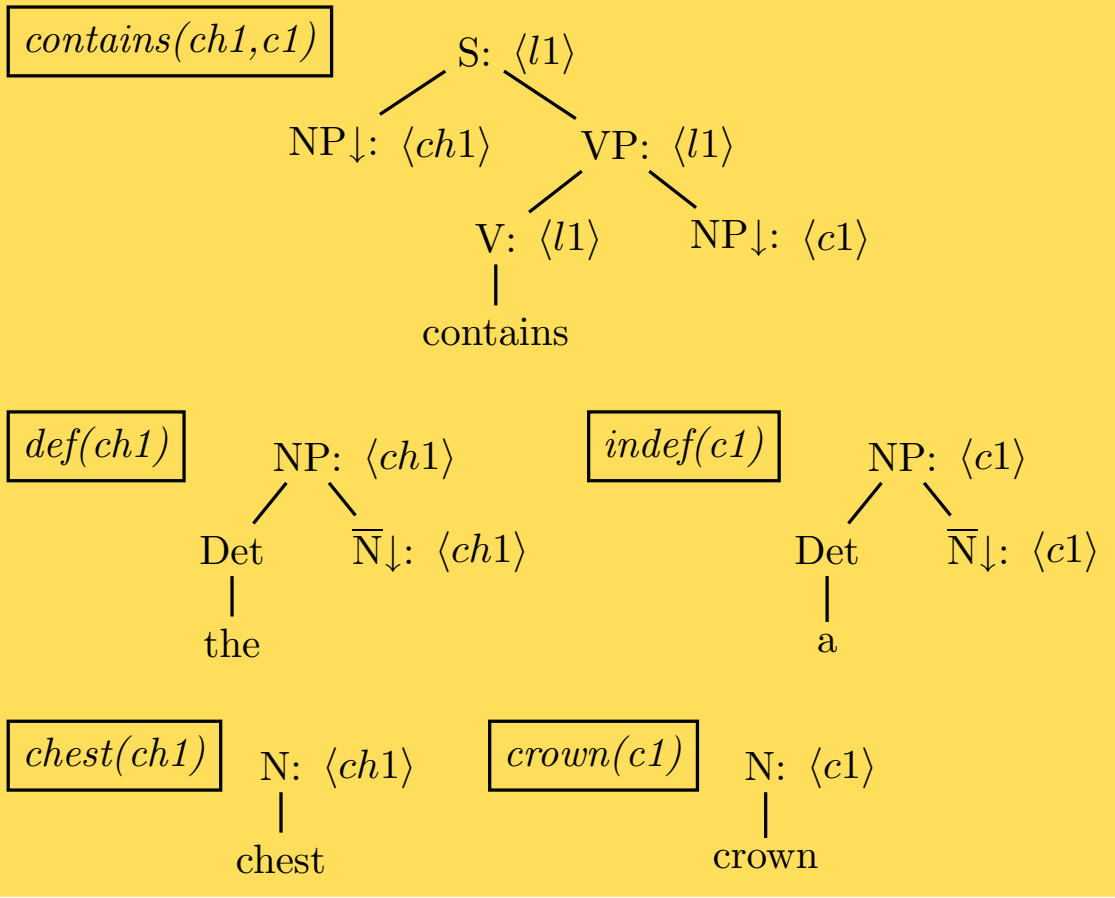
Realización del texto (Surface Realization)

$\{contains(ch1,c1), def(ch1), chest(ch1), indef(c1), crown(c1)\}$

Surface Realization

The chest contains a crown.

Gramática que asocia información semántica y pragmática con palabras y estructuras sintácticas.



Resumen: las tareas de generación

- Planificación del texto
  - Selección del contenido
  - Organización del contenido
- Planificación de las frases
  - Distribución del contenido en frases (agregación)
  - Generación de expresiones referenciales
  - Lexicalización
- Realización

Resumen: las tareas de generación

- Planificación del texto
  - Selección del contenido
  - Organización del contenido
- Planificación de las frases
  - Distribución del contenido en frases (agregación)
  - Generación de expresiones referenciales
  - Lexicalización
- Realización

La generación es más que el proceso inverso del parsing.

Posibles arquitecturas de sistemas de generación

Arquitecturas

Planificación del texto

- Selección del contenido
- Organización del contenido

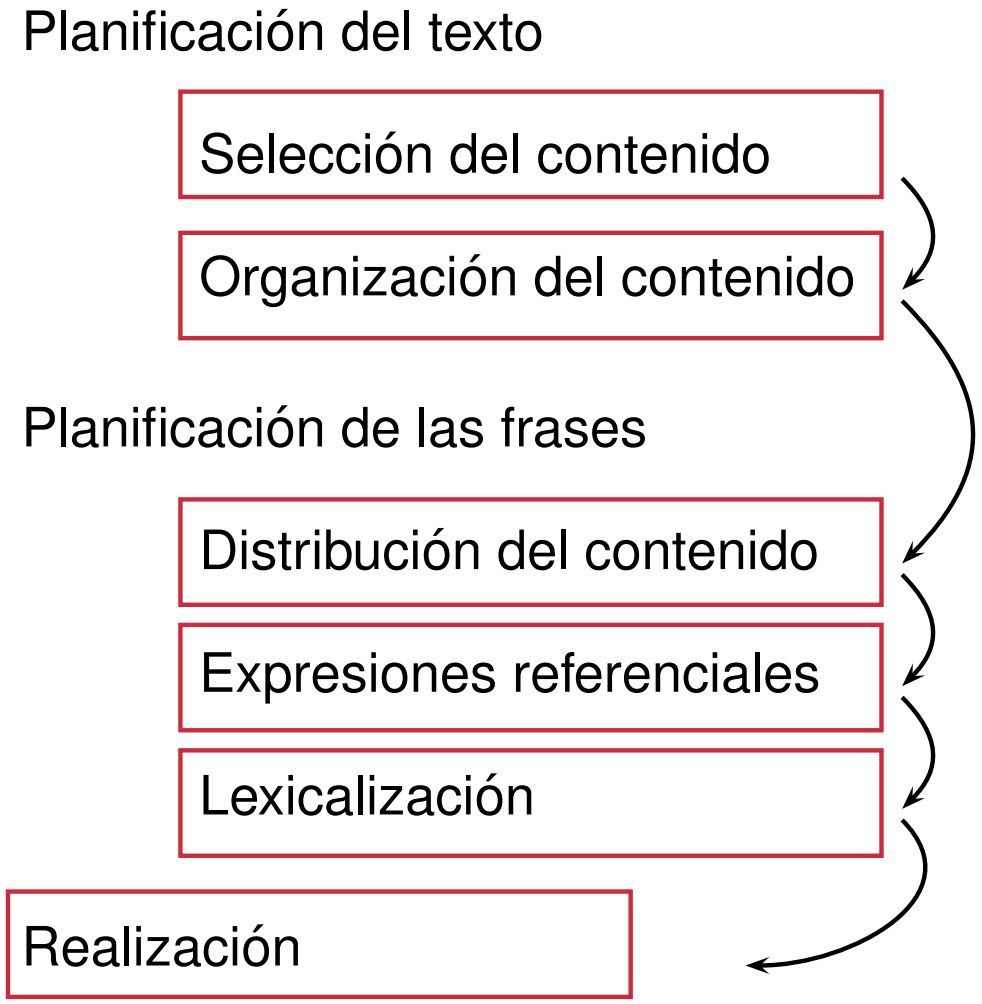
Planificación de las frases

- Distribución del contenido
- Expresiones referenciales
- Lexicalización

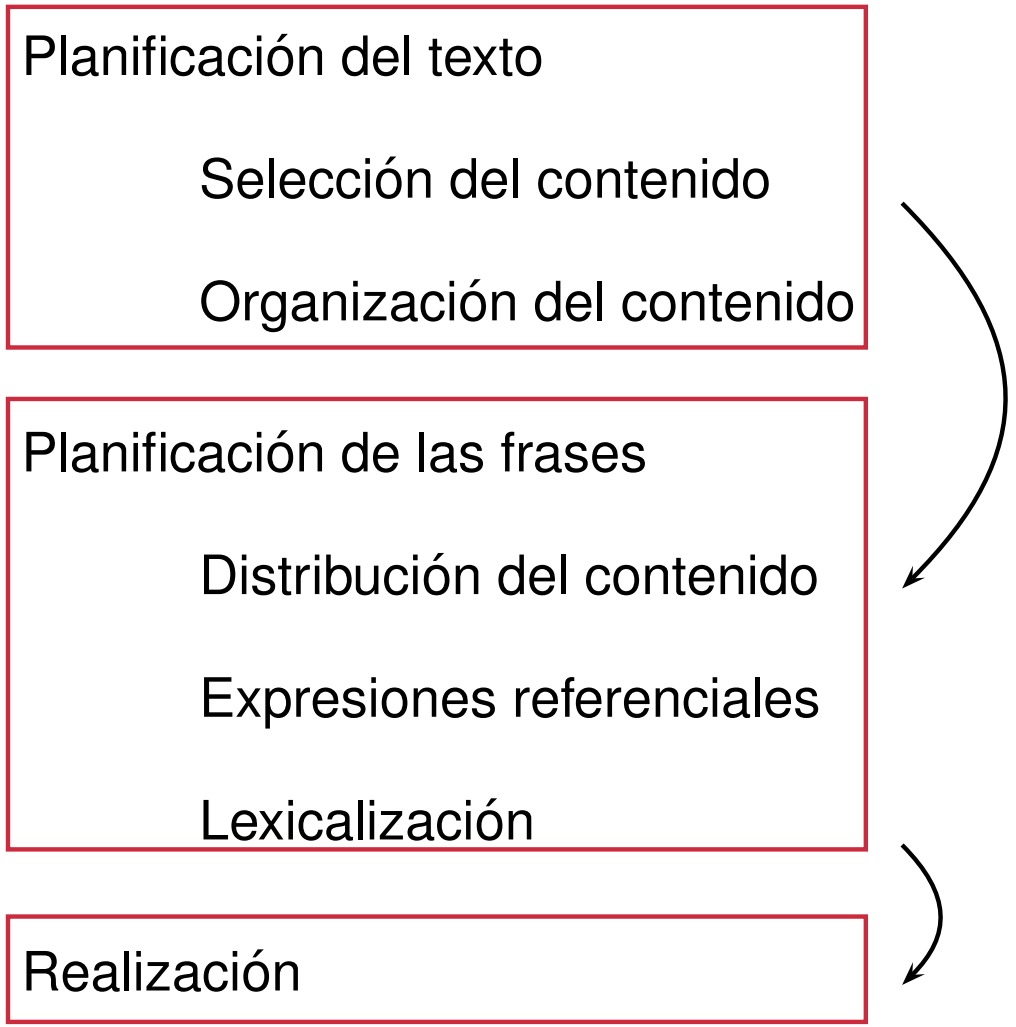
Realización

- Hay un consenso sobre las tareas que un sistema de generación tiene que tratar.
- Pero no hay consenso sobre el orden en que hay que tratarlas.
- Tampoco hay consenso sobre si las tareas pueden ser tratadas independientemente.

Tratamiento secuencial (1)



Tratamiento secuencial (2)



Sistemas integrados

Planificación del texto

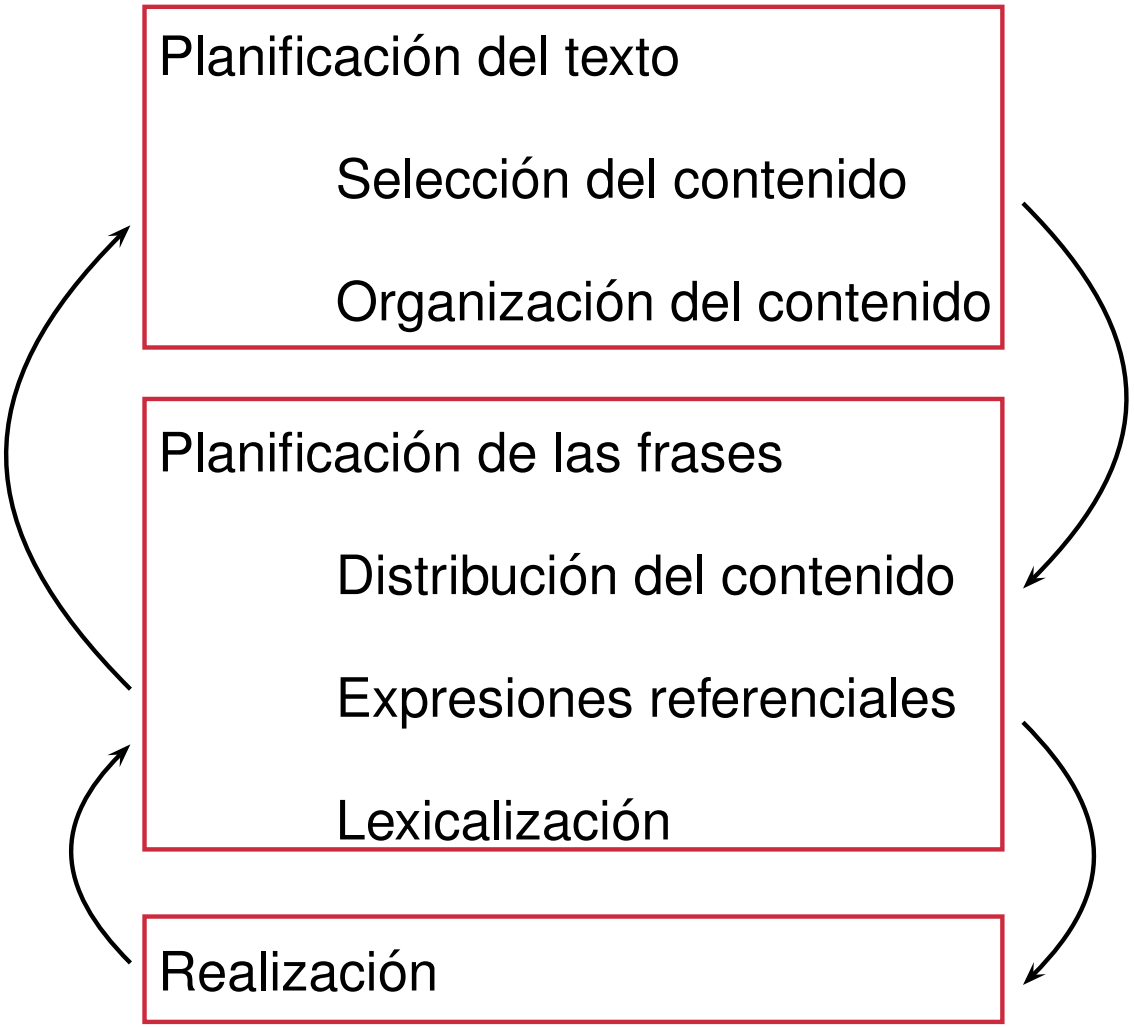
- Selección del contenido
- Organización del contenido

Planificación de las frases

- Distribución del contenido
- Expresiones referenciales
- Lexicalización

Realización

Sistemas modulares con “feedback”



Bibliographía

- ACL (Association of Computational Linguistics) mantiene una página de web que se llama “ACL Anthology” y donde hay versiones electrónicas de todas las actas de la conferencia anual de ACL, de las conferencias Coling, INLG (International Conference on Natural Language Generation) y otras conferencias en el área. También está la revista “Computational Linguistics”.
- panorama general, con énfasis en sistemas secuenciales, con muchas referencias

Reiter, E. and R. Dale (1997). Building Applied Natural Language Generation Systems. *Journal of Natural Language Engineering 3*, 57–87.

Reiter, E. and R. Dale (2000). *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

- otras arquitecturas

Stone, M. and C. Doran (1997). Sentence Planning as Description Using Tree Adjoining Grammar. In *Proceedings of the 35th Annual Meeting of the ACL*, pp. 198–205.

Stone, M., C. Doran, B. Webber, T. Bleam, and M. Palmer (2003). Microplanning with Communicative Intentions: The SPUD System. *Computational Intelligence 19*, 311–381.

Appelt, D. E. (1985a). Planning English Referring Expressions. *Artificial Intelligence 26*, 1–33.

Appelt, D. E. (1985b). *Planning English Sentences*. Cambridge University Press.

Rubinoff, R. (2000). Integrating Text Planning and Linguistic Choice Without Abandoning Modularity: The IGEN Generator. *Computational Linguistics 26*, 107–138.

- planificación del texto

  Hovy, E. (1988). Planning Coherent Multisentential Text. In *Proceedings of the 26th Annual Meeting of the ACL*, pp. 163–169.

  McKeown, K. (1985). *Text Generation*. Cambridge University Press.

  Moore, J. (1994). *Participating in Explanatory Dialogues*. Cambridge, MA: MIT Press.

  Moore, J. and C. Paris (1993). Planning Text for Advisory Dialogues. *Computational Linguistics 19*, 651–694.

- agregación

  Dalianis, H. (1999). Aggregation in Natural Language Generation. *Journal of Computational Intelligence 15*(4), 384–414.

  Shaw, J. (2002). *Clause Aggregation: An approach to generating concise text*. Ph. D. thesis, Columbia University.

- generación de expresiones referenciales

  Dale, R. and E. Reiter (1995). Computational Interpretations of the Gricean Maxims in the Generation of Referring Expressions. *Cognitive Science 18*, 233–263.

  Passonneau, R. J. (1996). Using Centering to Relax Gricean Informational Constraints on Discourse Anaphoric Noun Phrases. *Language and Speech 39*(2–3), 229–264.

- realización

  Brew, C. (1992). Letting the Cat out of the Bag: Generation for Shake-and-bake MT. In *Proceedings of the 15th Coling*, pp. 610–616.

  Kay, M. (1996). Chart Generation. In *34th Annual Meeting of the ACL*, pp. 200–204.

Carroll, J., A. Copestake, D. Flickinger, and V. Poznanski (1999). An Efficient Chart Generator for (Semi-)Lexicalist Grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*, pp. 86–95.

Koller, A. and K. Striegnitz (2002). Generation as Dependency Parsing. In *Proceedings of the 40th Annual Meeting of the ACL*, pp. 17–24.

- generación con gramáticas funcionales

Batemann, J. KPML one-point acccess page. `http://purl.org/net/kpml`

Halliday, M. A. (1985). *An Introduction to Functional Grammar*. London: Edward Arnold.

- otros temas
  - generación de entonación

Prevost, S. (1996). An Information Structural Approach to Spoken Language Generation. In *Proceedings of the 34th Annual Meeting of the ACL*, pp. 294–301.

  - generación de gestos

Cassell, J., M. Stone, and H. Yan (2000). Coordination and context-dependence in the generation of embodied conversation. In *Proceedings of the First International Conference on Natural Language Generation (INLG)*, pp. 171–178.

- un artículo sobre el text adventure

Koller, A., R. Debusmann, M. Gabsdil, and K. Striegnitz (2004). Put my galakmid coin into the dispenser and kick it: Computational Linguistics and Theorem Proving in a Computer Game. *Journal of Logic, Language and Information (JoLLI) 13*, 187–206.