# Generierung mit SPUD

SPUD steht für “Sentence Planning Using Description”

SPUD ist von Matthew Stone entwickelt worden. (`www.cs.rutgers.edu/~mdstone`)

SPUD ist ein Beispiel, wie der Unterschied zwischen ‘Assertion’ und ‘Presupposition’ in einem Generierungssystem verwendet werden kann.

# Verschiedene Sichtweisen auf Präsuppositionen (1)

1. Präsuppositionen sind “komische” logische Schlussfolgerungen

*“The king of France is bald”* präsupponiert, dass es einen König von Frankreich gibt.

*“It is not true that the king of France is bald”* präsupponiert ebenfalls, dass es einen König von Frankreich gibt.

# Verschiedene Sichtweisen auf Präsuppositionen (2)

2. Präsuppositionen sind Anaphern (van der Sandt)

| r g e |
| --- |
| rabbit(r) small(r)<br>garden(g) poss(speaker, g)<br>sit_in(e, r, g) |

*“A small rabbit is sitting in my garden.”*

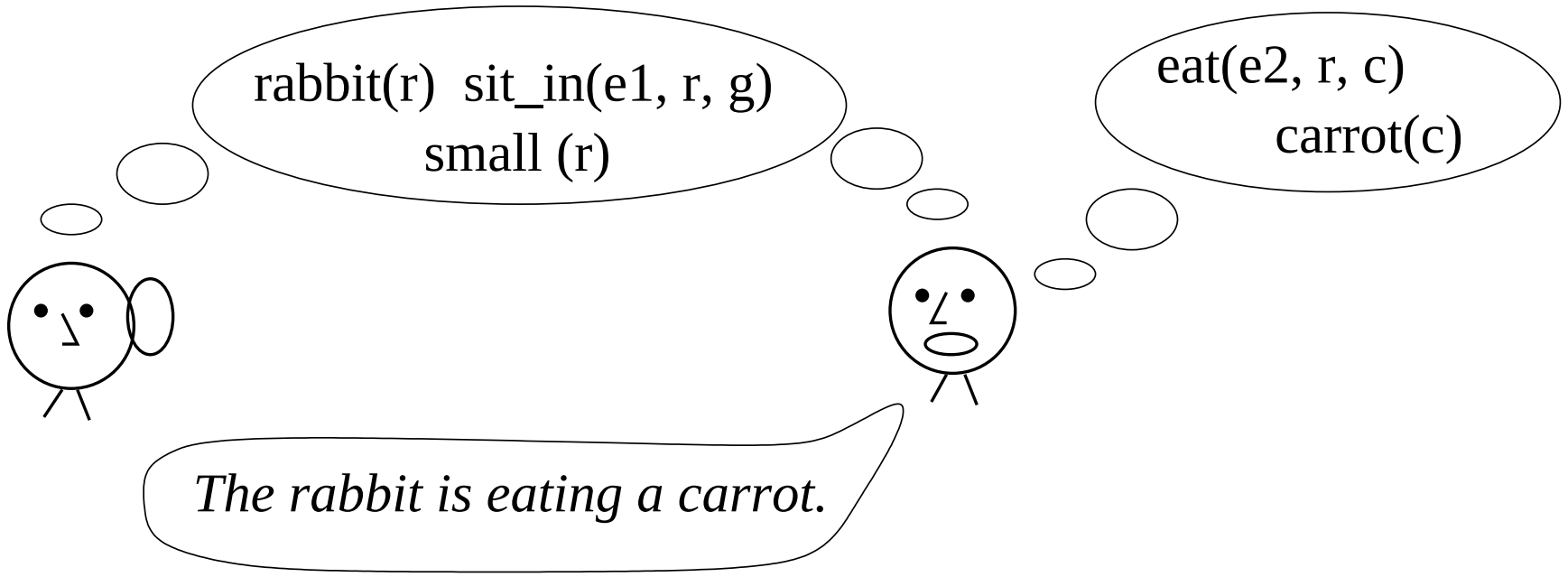
| r g e c e2 |
| --- |
| rabbit(r) small(r)<br>garden(g) poss(speaker, g)<br>sit_in(e, r, g)<br>carrot(c) eat(e2, r, c) |

*“A small rabbit is sitting in my garden.”*

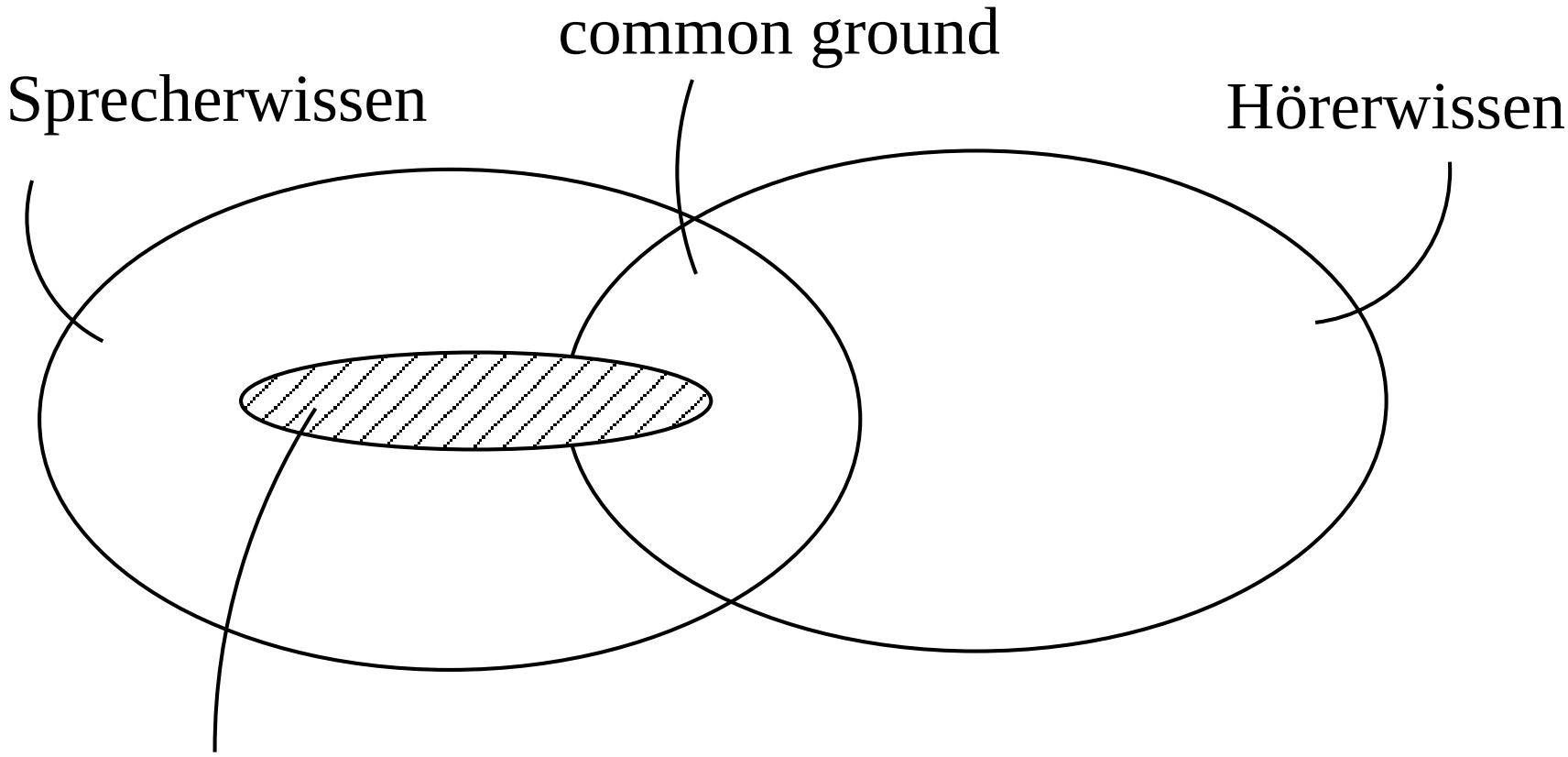
*“The rabbit is eating a carrot.”*

# Verschiedene Sichtweisen auf Präsuppositionen (3)

3. Präsuppositionen sagen dem Hörer, wie die Bedeutung des Satzes mit bereits bekannten Individuen und Tatsachen zusammenhängt.



# SPUDs Bild von Kommunikation




Bem.: Die Möglichkeit von Akkomodation wird ignoriert.

# SPUDs Aufgabenstellung

Gegeben:

Kontext:

- Wissen des Sprechers/Systems
- Wissen, das System und Hörer teilen
- Informationen über die Diskursgeschichte und -struktur; z.B. Informationen über die Salienz von Individuen, offenen Fragen ...

Kommunikationsziel:

- eine syntaktische Kategorie $C$
- ein (semantisches) Individuum $a$
- eine Menge von Fakten $\Gamma$

Aufgabe: Konstruiere eine Äußerung der Kategorie $C$, die im vorgegebenen Kontext das Individuum $a$ beschreibt und alle Fakten aus $\Gamma$ ausdrückt.

# Beispiel

| shared | private |
|---|---|
| rabbit($r_1$) rabbit($r_2$) | be_in($s_1$, $r_1$, $h_1$) |
| white($r_1$) black($r_2$) | hat($h_1$) |
| has($s_2$, $r_2$, $t$) tail($t$) fluffly($t$) | |
| status($r_1$, in_focus) | |

Ziel: $\langle$ NP, $r_2$, $\emptyset$ $\rangle$

$\longrightarrow$ *“the black rabbit”* or *“the rabbit with the fluffy tail”* or *“the black rabbit with the fluffy tail”*

Ziel: $\langle$ NP, $r_2$, $\{$ has($s_2$, $r_2$, $t$), tail($t$), fluffly($t$)$\}$ $\rangle$

$\longrightarrow$ *“the rabbit with the fluffy tail”* or *“the black rabbit with the fluffy tail”*

Ziel: $\langle$ S, $s_1$, $\emptyset$ $\rangle$

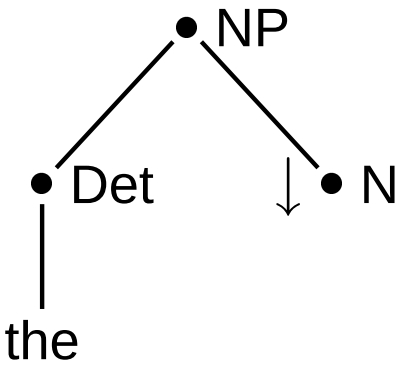
$\longrightarrow$ *“the white rabbit is in a hat”*

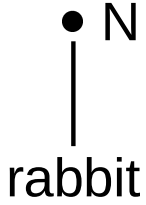
# SPUDs Grammatik

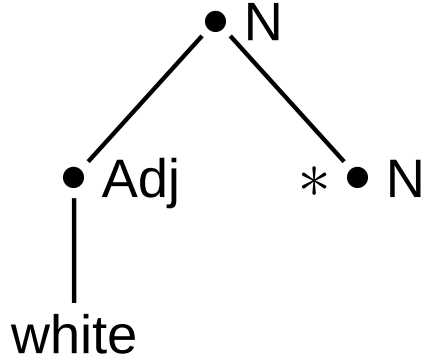
- (Lexicalized) Tree Adjoining Grammar —(L)TAG
- Elementarbäume sind mit semantischer und pragmatischer Information assoziert

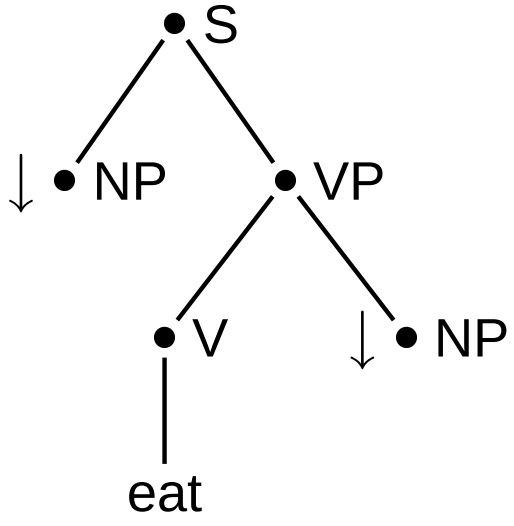
# Exkurs: TAG (1)

Eine TAG-Grammatik ist eine Menge von Bäumen, sogenannten Elementarbäumen.



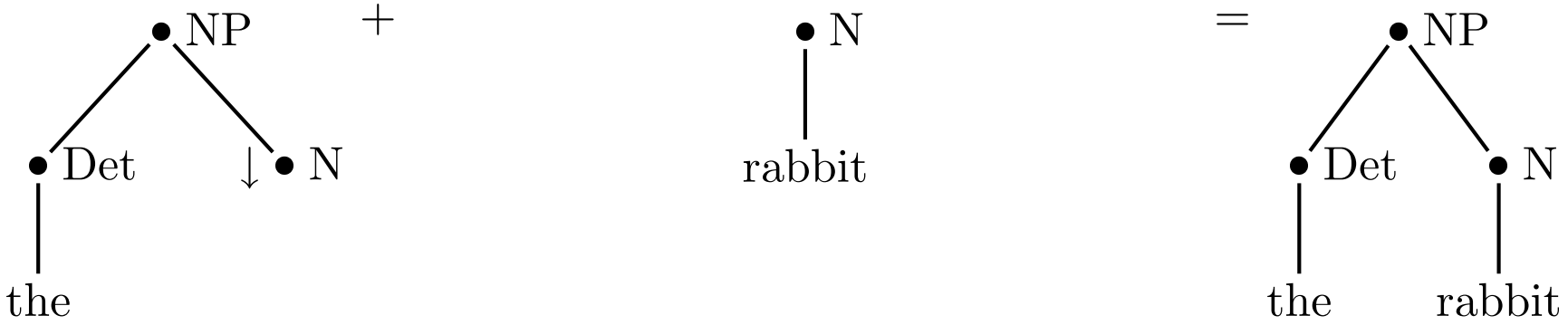




(entwickelt von Aravind Joshi und vielen Anderen)

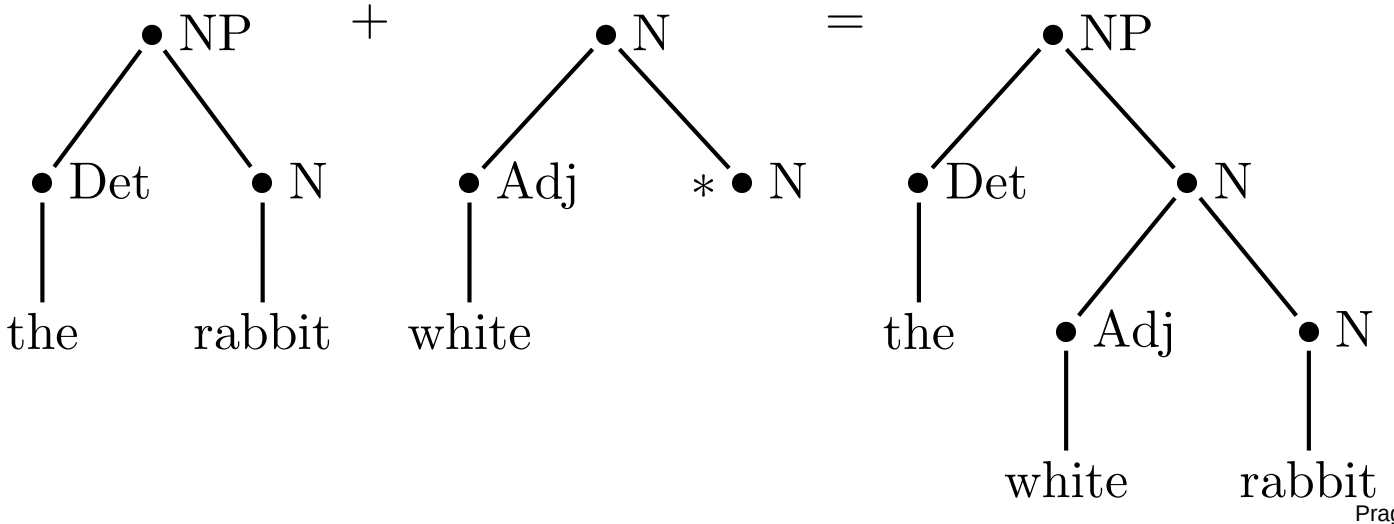
# Exkurs: TAG (2)

Es gibt zwei Operationen, mit denen Bäume “zusammengebaut” werden können.

Substitution:

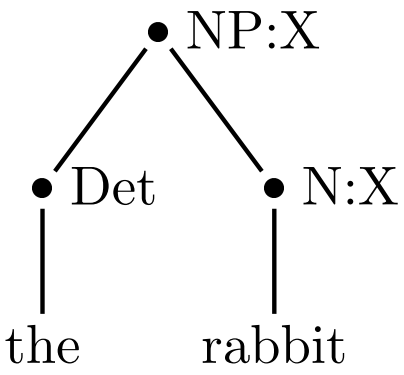



Adjunktion:

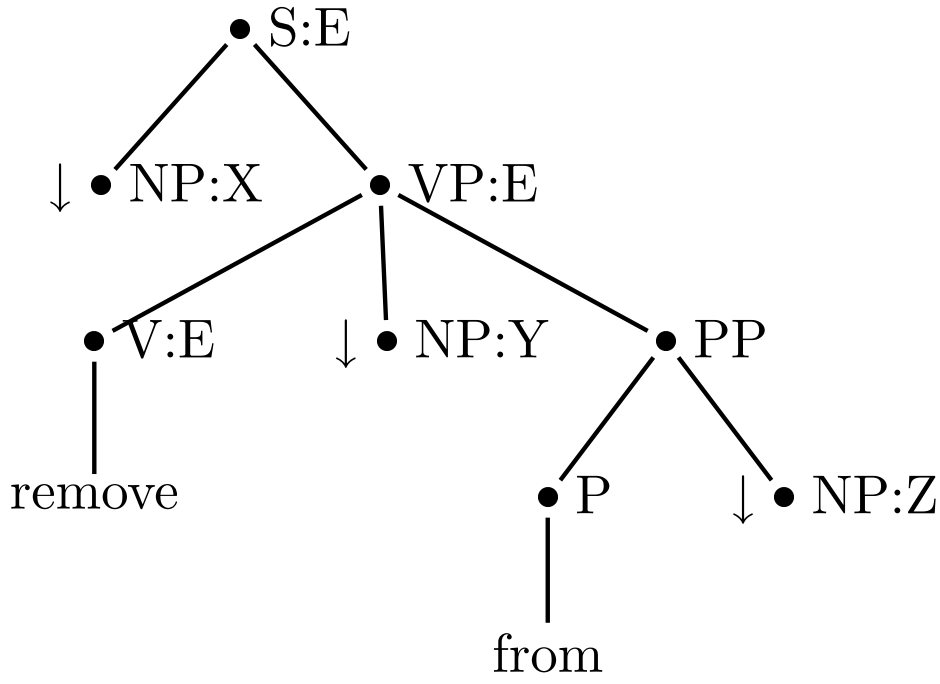


# SPUDs Grammatik

- (Lexicalized) Tree Adjoining Grammar —(L)TAG
- Elementarbäume sind mit semantischer und pragmatische Information assoziert




semantics:
assertion: ∅
presupposition: { rabbit(X) }
pragmatics:
{ uniquely_identifiable(X) }




semantics:
assertion: { remove(E, X, Y, Z)}
presupposition: { is_in(Y, Z)}
pragmatics: ∅

# Adjunktion und Substitution in SPUD

- Auf der syntaktischen Ebene funktionieren Adjunktion und Substitution wie in “normalen” TAG.
- Die semantische und pragmatische Information wird vereinigt.
- Die Assertion muss wahr sein bezüglich des Sprecherwissens.
- Die Präsupposition muss wahr sein bezüglich des gemeinsamen Wissens.
- Die pragmatischen Constraints müssen erfüllt sein bezüglich der Diskursgeschichte.

# SPUDs Suchalgorithmus

Kommunikationsziel: $\langle\, C, a, \Gamma \,\rangle$

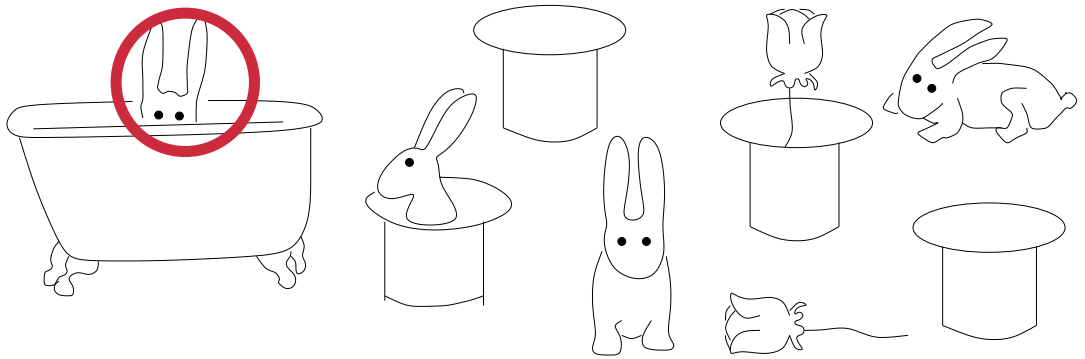
1. Initialisiere den TAG-Baum mit $\bullet\ \langle C, a \rangle$.
2. Falls der TAG-Baum
   - syntaktisch vollständig ist
   - alle Fakten aus $\Gamma$ ausdrückt
   - alle präsupponierten Individuen eindeutig beschreibt,

   dann gib diesen TAG-Baum als Lösung zurück.
3. Ansonsten berechne alle Möglichkeiten, wie der Baum erweitert werden kann.
4. Wähle die beste Möglichkeit und mache bei Schritt 2 weiter.

(SPUD benutzt eine greedy-search Strategie.)

# Beispiel

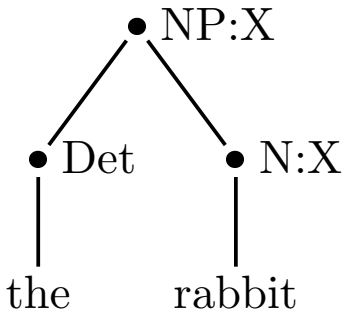
common ground:



Ziel: $\langle$ NP, rab1, $\emptyset$ $\rangle$

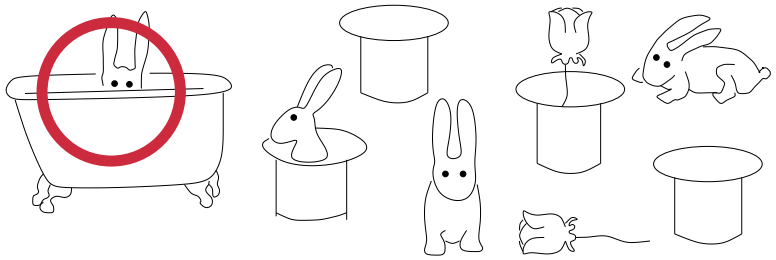
$\downarrow \bullet$ NP:rab1

semantics:
assertion: $\emptyset$
presupposition: $\emptyset$
pragmatics: $\emptyset$

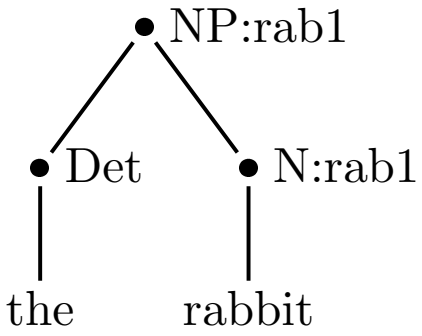



semantics:
assertion: $\emptyset$
presupposition: { rabbit(X) }
pragmatics:
{ uniquely_identifiable(X) }

# Beispiel



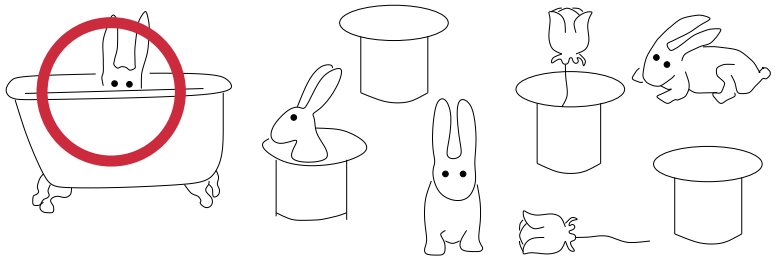
Ziel: $\langle$ NP, rab1, $\emptyset$ $\rangle$



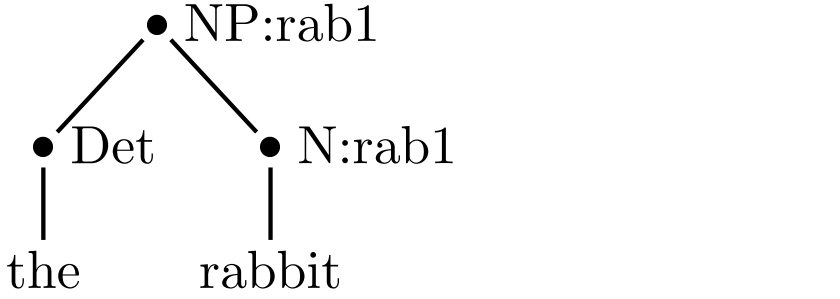

semantics:
assertion: $\emptyset$
presupposition: { rabbit(rab1) }
pragmatics:
{ uniquely_identifiable(rab1) }

- rabbit(rab1) ist wahr bzgl. des common ground: YES
- uniquely_identifiable(rab1) ist wahr bzgl. des common ground: YES
- der Baum hat keine offenen Substitutionsknoten: YES
- rab1 ist eindeutig beschrieben: NO

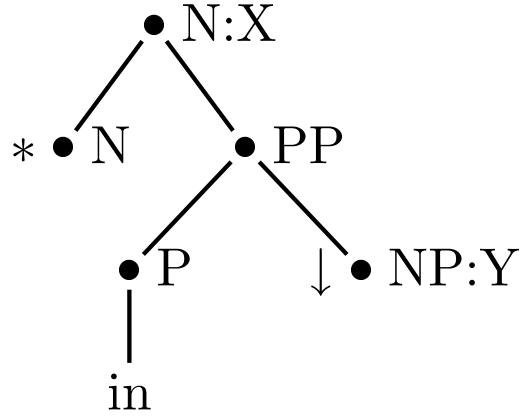
# Beispiel



Ziel: $\langle$ NP, rab1, $\emptyset$ $\rangle$

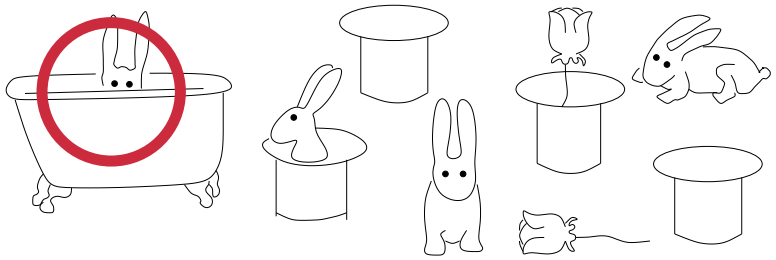



semantics:
a: $\emptyset$
p: { rabbit(rab1) }
pragmatics:
{ uniquely_identifiable(rab1) }

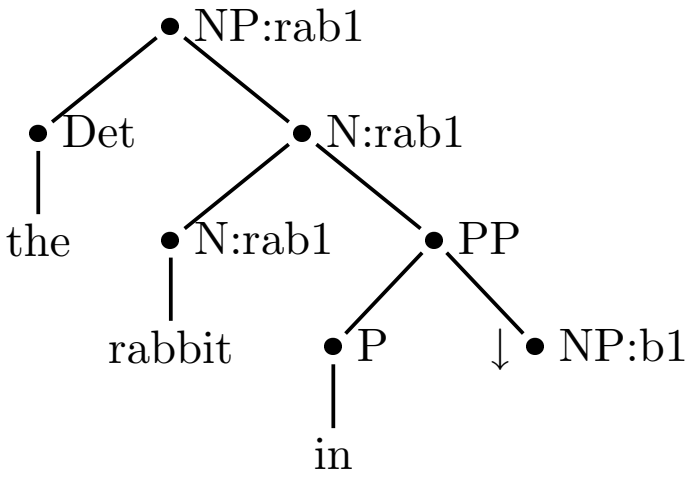



semantics:
a: $\emptyset$
p: { is_in(X,Y) }
pragmatics: $\emptyset$

# Beispiel



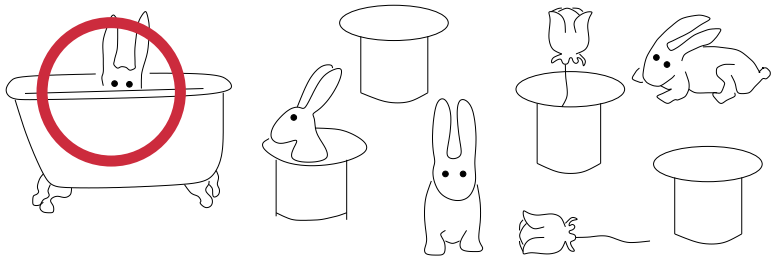
Ziel: $\langle$ NP, rab1, $\emptyset$ $\rangle$



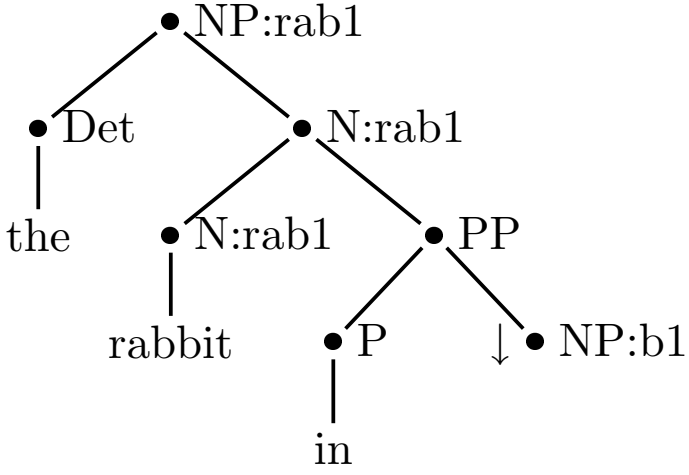

semantics:
a: $\emptyset$
p: { rabbit(rab1), is_in(rab1, b1) }
pragmatics:
{ uniquely_identifiable(rab1) }

- is_in(rab1, b1) ist wahr bzgl. des common ground: YES
- der Baum hat keine offenen Substitutionsknoten: NO
- rab1 ist eindeutig beschrieben: NO

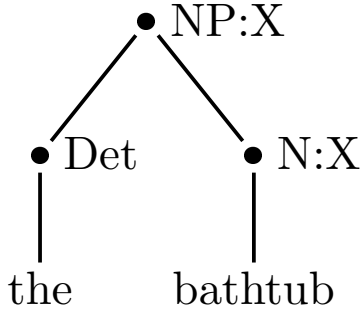
# Beispiel



Ziel: $\langle$ NP, rab1, $\emptyset$ $\rangle$



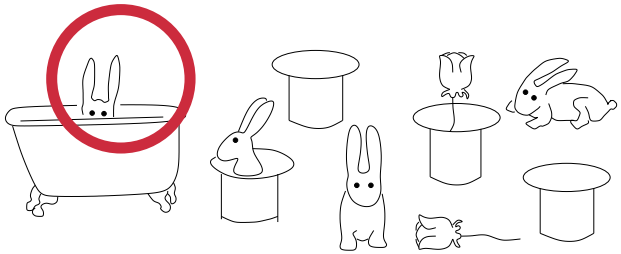

semantics:
a: $\emptyset$
p: { rabbit(rab1), is_in(rab1, b1) }
pragmatics:
{ uniquely_identifiable(rab1) }

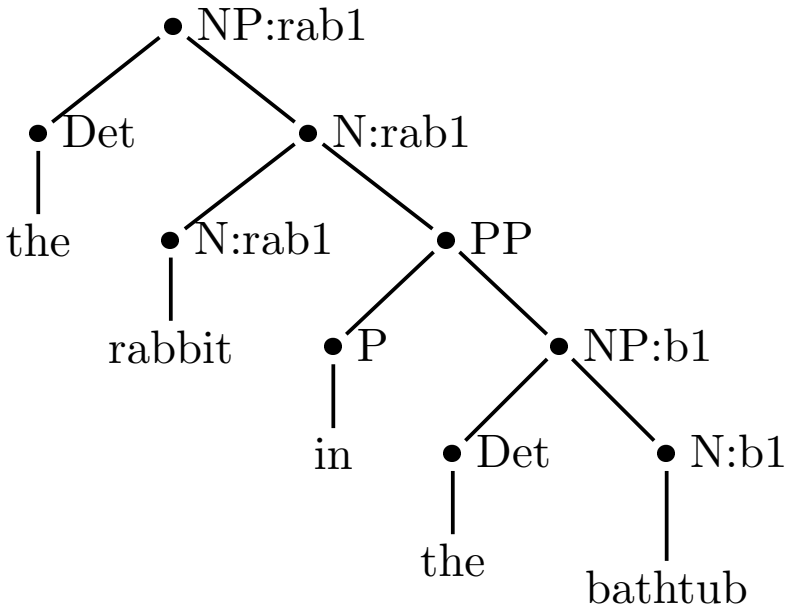



semantics:
assertion: $\emptyset$
presupposition: { bathtub(X) }
pragmatics:
{ uniquely_identifiable(X) }

# Beispiel

Ziel: $\langle$ NP, rab1, $\emptyset$ $\rangle$






semantics:
a: $\emptyset$
p: { rabbit(rab1), is_in(rab1, b1),
bathtub(b1) }
pragmatics:
{ uniquely_identifiable(rab1),
uniquely_identifiable(b1) }

- bathtub(b1) ist wahr bzgl. des common ground: YES
- uniquely_identifiable(b1) ist wahr bzgl. des common ground: YES
- der Baum hat keine offenen Substitutionsknoten: YES
- rab1 ist eindeutig beschrieben: YES
- b1 ist eindeutig beschrieben: YES

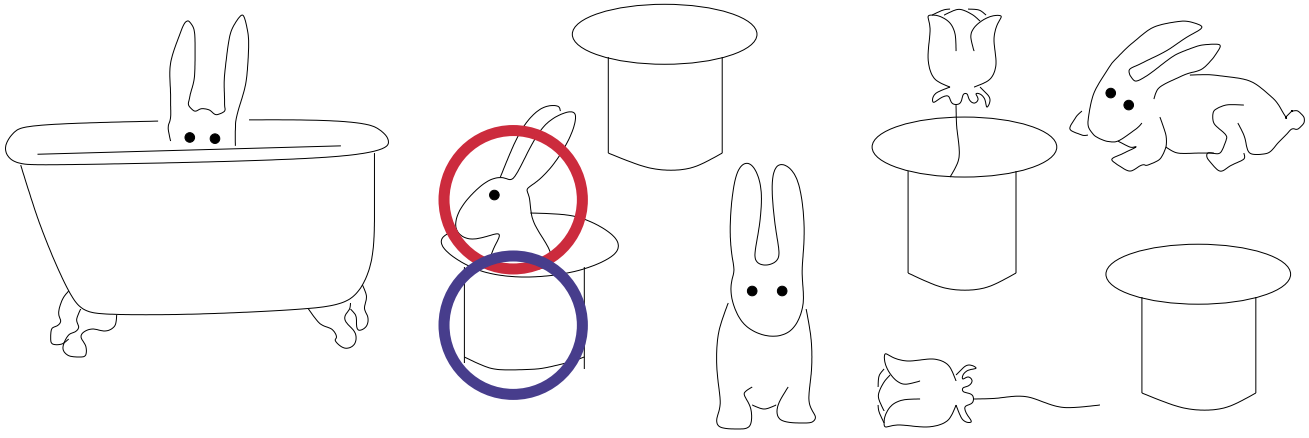
# Heuristiken, die die Suche steuern

Falls es mehrer Möglichkeiten gibt, den TAG-Baum zu erweitern, nutze die folgenden Kriterien um Möglichkeiten zu eliminieren, bis eine einzige Möglichkeit übrigbleibt.

1. Behalte nur die Bäume, die die meisten der im Kommunikationsziel angegebenen Fakten ausdrücken.
2. Behalte nur die Bäume, die die präsupponierten Individuen am besten beschreibt.
3. Behalte nur die Bäume, die am meisten saliente und am wenigsten nicht-saliente Entitäten erwähnen.
4. Behalte nur die Bäume mit den wenigsten offenen Substitutionsknoten.
5. Behalte nur die Bäume, die die spezifischsten pragmatischen Constraints haben.

# ‘Textual Economy’ (1)

common ground:



Ziel: $\langle$ NP, rab2, $\emptyset$ $\rangle$

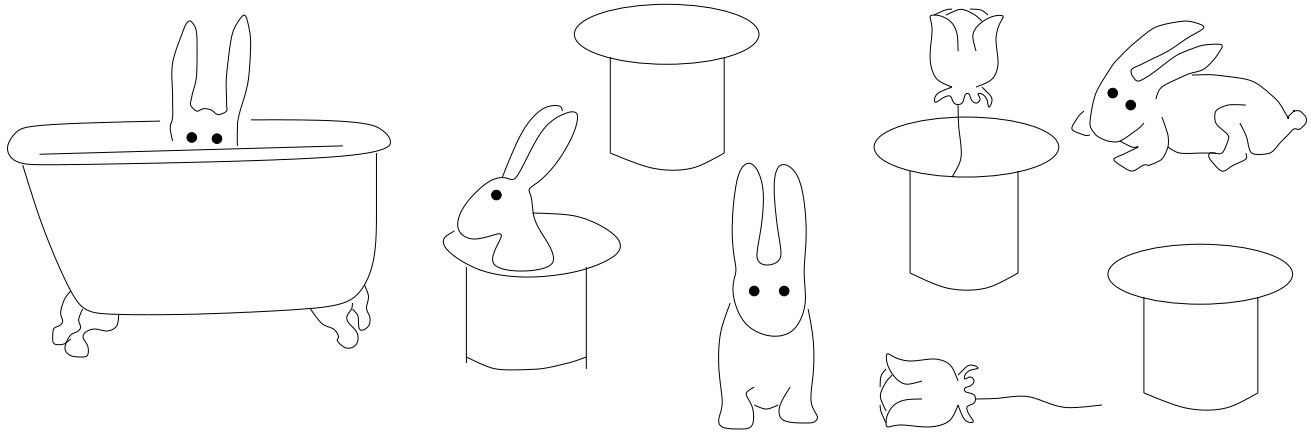
$\longrightarrow$ *the rabbit in the hat*

Ziel: $\langle$ NP, hat1, $\emptyset$ $\rangle$

$\longrightarrow$ *the hat with the rabbit*

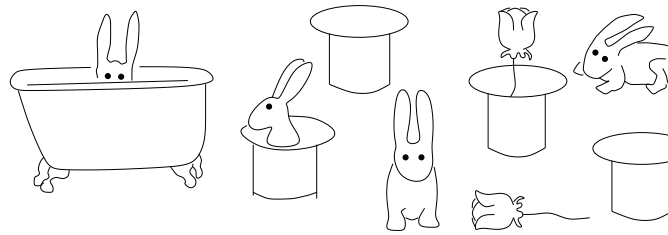
# ‘Textual Economy’ (2)

common ground:



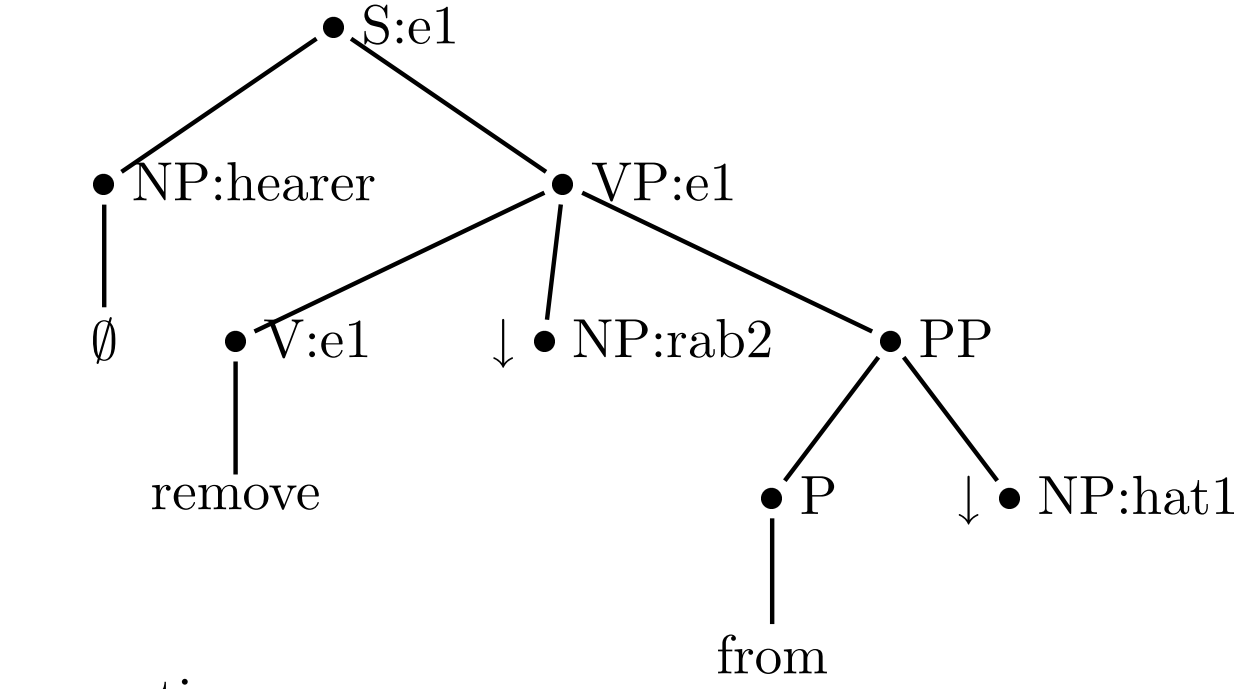
Ziel: $\langle$ S, e1, $\{$ remove(e1, hearer, rab2, hat1), do_next(e1)$\}$ $\rangle$

# ‘Textual Economy’ (2)



Ziel: ⟨ S, e1, { remove(e1, hearer, rab2, hat1),

do_next(e1)} ⟩



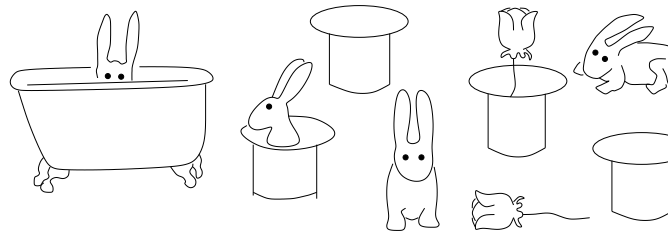

semantics:

assertion: { remove(e1, hearer, rab2, hat1), do_next(e1)}

presupposition: { is_in(rab2, hat1)}

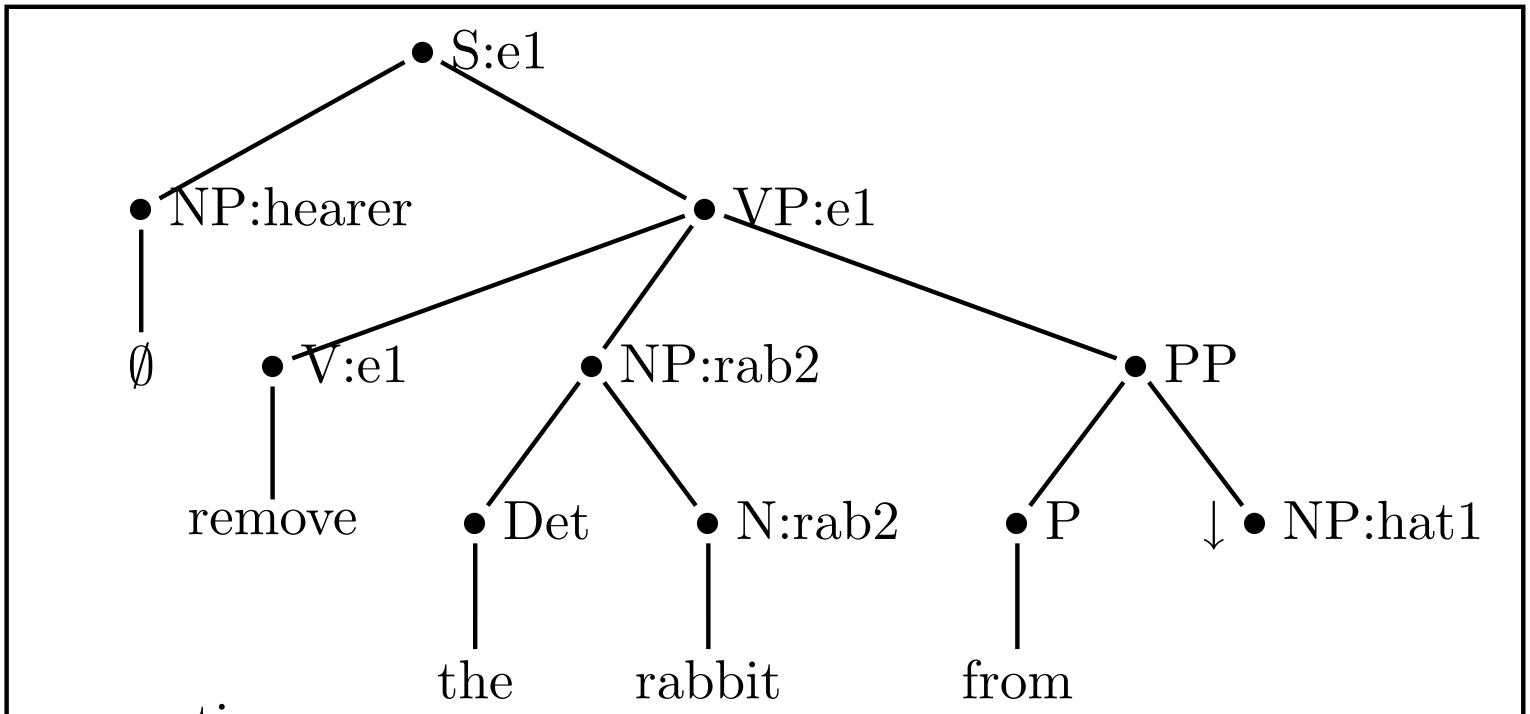
pragmatics: ∅

# ‘Textual Economy’ (2)



Ziel: ⟨ S, e1, { remove(e1, hearer, rab2, hat1),

do_next(e1)} ⟩



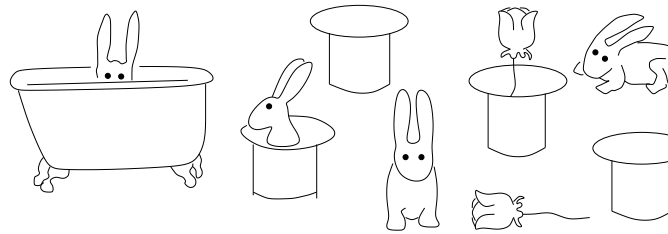

semantics:

a: { remove(e1, hearer, rab2, hat1), do_next(e1)}

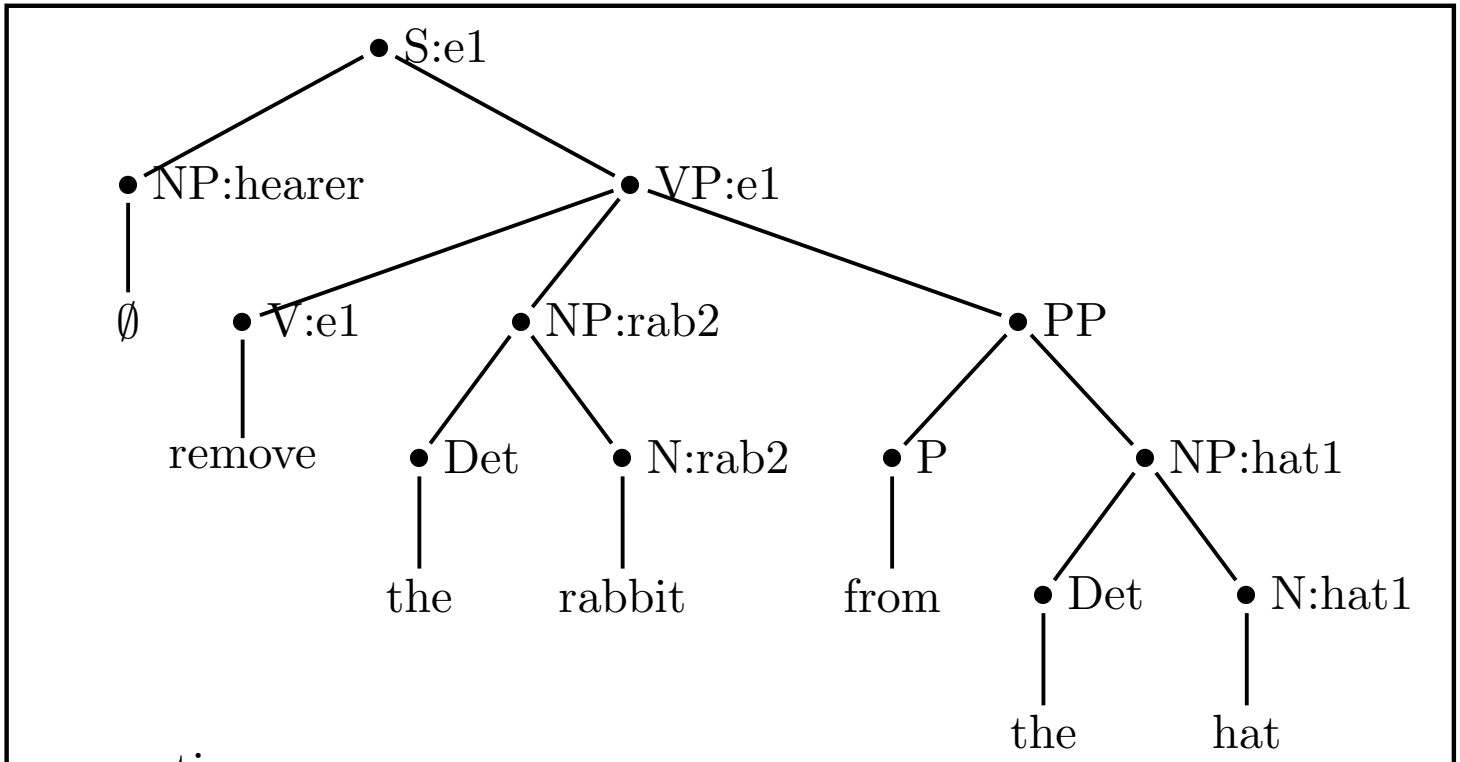
p: { is_in(rab2, hat1), rab(r2) }

pragmatics: { uniquely_identifiable(rab2) }

# ‘Textual Economy’ (2)



Ziel: ⟨ S, e1, { remove(e1, hearer, rab2, hat1), do_next(e1)} ⟩




semantics:
a: { remove(e1, hearer, rab2, hat1), do_next(e1)}
p: { is_in(rab2, hat1), rab(r2), hat(h1) }
pragmatics: { uniquely_identifiable(rab2),
uniquely_identifiable(hat1) }

# Zusammenfassung

- SPUD macht Satzplanung und Oberflächenrealisierung gleichzeitig (anders als in der Standardarchitektur von vor zwei Wochen).
- Wörter und syntaktische Strukturen sind mit semantischer und pragmatischer Information assoziiert.
- Die pragmatische Information beschreibt, in welchen Diskurssituationen der Ausdruck verwendet werden darf.
- Die semantische Information wird in Assertion und Präsupposition unterteilt. Assertion: die (neue) Information, die der Sprecher dem Hörer mitteilen will; Präsupposition: verankert die Bedeutung des Satzes im common ground.
- SPUD unterstüzt die Generierung von “ökonomischen” Äußerungen, wie z.B. *remove the rabbit from the hat* anstatt *remove the rabbit in the hat from the hat with the rabbit*.

# SPUD in Prolog

- taglet.pl (`www.cs.rutgers.edu/~mdstone/class/taglet/`) stellt eine Reihe von Prädikaten zur Verarbeitung von (einer Variante von) TAGs bereit.
- SPUDs Suchalgorithmus ist auch implementiert (spud_search/2).
- Eine kleine Grammatik ist dabei.
- Der Context muss selber definiert werden. Z.B.

```
shared(type(d1, dog)).
shared(type(c1, cat)).
shared(type(d2, dog)).
shared(color(s1, d1, white)).
shared(color(s2, d2, black)).
shared(status(d1, the)).
shared(status(d2, the)).
private(sleeping(e1)).
private(sleeper(e1, d1)).
```

?- spud_ref(d2, T).

$\longrightarrow$ generiert eine NP, die d2 beschreibt

?- spud_search(initial_state(s(_), s(e1), []), T).

$\longrightarrow$ startet die SPUD-Suche mit dem Kommunikationsziel $\langle$ S, e1, $\emptyset$ $\rangle$