LING 364: Introduction to Formal Semantics

Lecture 27

April 27th

Administrivia

- **Homework 5**
  - **all returned**
    - *if you didn't get an email from me,*
    - *I didn't get your homework*

Administrivia

- **Homework 6**
  - short homework on time and tense
  - out today
  - due to proximity to the end of the semester and the final
  - **due next Tuesday**

Homework 6 help: come to my office

Availability
tomorrow (Friday) (whole afternoon)
Monday(whole afternoon)
Tuesday (last lecture)

Administrivia

- **Final**
  - a take-home
  - out next Tuesday
  - **you have one day+**
  - **due that Wednesday**
    - **I will be available all day Wednesday for questions**
    - **(Douglass 308)**

Time and Tense

- Recap of formal concepts:
  - **(S) utterance** or **speech time**
  - **(E) event time**
  - **(T) reference (R)** or **topic time**
  - time **intervals**
    - the notion that E,S and T are intervals
  - interval **relations**:
    - precedence (<)
    - inclusion (⊆)

A Grammar for Tense and Time

- sbar(R) --> adjunct(R1), s(R2), {append(R1,R2,R)}.
- sbar(R) --> s(R).

- s(R) --> np, vp(R).
- np --> [i].
- np --> [noah].
- vp(R) --> v(R1,go), [for,a,hike], {append([(subset(e,t))],R1,R)}.
- vp(R) --> v(R1,have), [a,rash], {append([intersect(e,t)],R1,R)}.

- v([(t<s)],go) --> [went].
- v([(t=s)],go) --> [go].
- v([(s<t)],go) --> [will,go].

- v([(t<s)],have) --> [had].
- v([(t=s)],have) --> [have].
- v([(s<t)],have) --> [will,have].

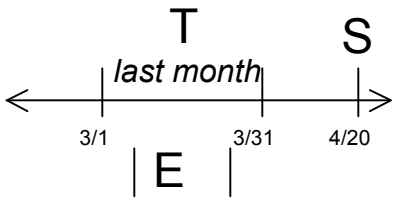
- adjunct([(t<s),t=last_month(s)]) --> [last,month].
- adjunct([(t<s),t=yesterday(s)]) --> [yesterday].
- adjunct([(s=t),t=today(s)]) --> [today].
- adjunct([(s<t),t=tomorrow(s)]) --> [tomorrow].

simple grammar we will use for the homework

a more elaborate grammar would integrate, i.e. include,
the meaning grammars
that we've been developing in other homework

Exercise 1

- Let's see what this grammar computes
- Run
  - (16) **Last month**, I **went** for a hike
- as follows
  - ?- sbar(R,[last,month,i,went,for,a,hike],[]).
  - R = [t<s,t=last_month(s),subset(e,t),t<s]



Exercise 1

- **Explaining the output**
  - ?- sbar(R,[last,month,i,went,for,a,hike],[]).
  - R = [**t<s,t=last_month(s)**,subset(e,t),t<s]
  - *Each part of the sentence that has something to say about time/tense contributes some part of the result*
  - *each part, e.g. R, R1, R2, is stored as a Prolog list*

Relevant Grammar Rules

•sbar(**R**) --> adjunct(**R1**), s(R2), {append(**R1**,R2,**R**)}.

•**remember**: append/3 concatenates lists R1 and R2 to make R

–let's look at R1 which comes from the rule for adjunct

•adjunct([(t<s),t=last_month(s)]) --> [last,month].

•R1 = [(t<s),t=last_month(s)]

•list containing two facts

•t < s (reference time T precedes utterance time S)

•t = last_month(s)

Exercise 1

- **Explaining the output**
  - ?- sbar(R,[last,month,i,went,for,a,hike],[]).
  - R = [t<s,t=last_month(s),**subset(e,t),t<s**]

Relevant Grammar Rules

•sbar(R) --> adjunct(R1), s(**R2**), {append(R1,R2,R)}.

–let's look at the 2nd half of the result

–**R2** comes from the rule for S

•s(**R**) --> np, vp(**R**).

•np --> [i].

•vp(**R**) --> v(**R1**,go), [for,a,hike], {append(**[subset(e,t)],R1**,**R**)}.

–subset(e,t) encodes E ⊆T

–aspectual information: "go for a hike" is an **accomplishment,** and

–happens in the reference time interval

•v(**[(t<s)]**,go) --> [**went**].

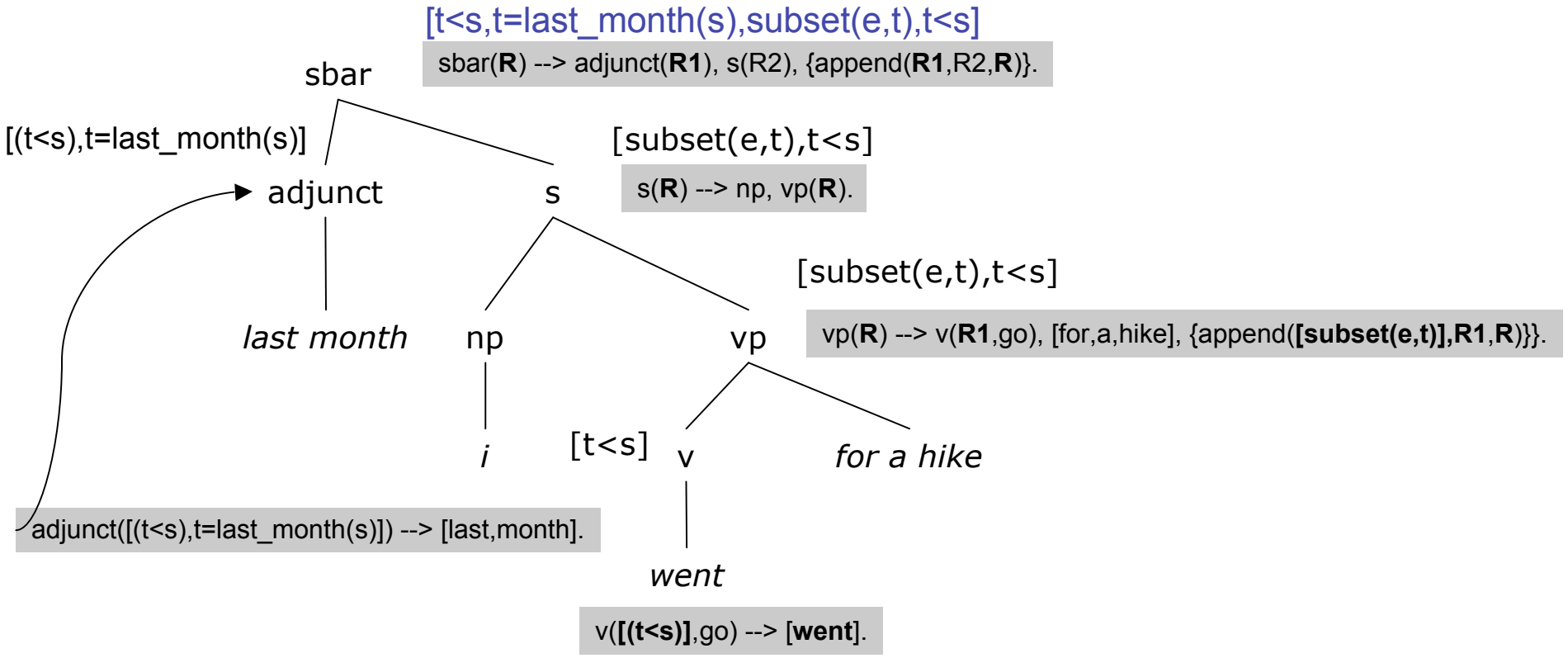
•v([(s<t)],go) --> [will,go].

–**R1** = [(t<s)]

–t<s encodes past tense, i.e. T < S

Exercise 1

- In diagram form:



Exercise 1

- **An inference rule**
  - infer(R,[(Z<Y)]) :-
  - select((X<Y),R,R1),
  - select(subset(Z,X),R1,_).
  - *% select(X,L,L')*
  - *% selects X a member of list L,*
  - *% L' is the list L with X removed*
  - select(X,[X|L],L).
  - select(X,[Y|L],[Y|Lp]) :- select(X,L,Lp).

- **Encodes**:
  - If
  - $X < Y$
  - and
  - $Z \subseteq X$
  - we can infer:
  - $Z < Y$
- over the list of relations given in R

Exercise 1

- Running
  - ?- sbar(R,[last,month,i,went,for,a,hike],[]).
  - R = [t<s,t=last_month(s),**subset(e,t),t<s**]

- What should I be able to infer?
  - **Answer**: E < S

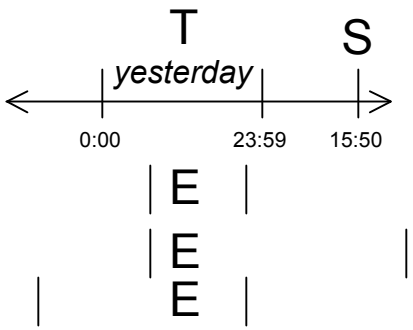
- Let's use our inference rule!
  - ?- sbar(R,[last,month,i,went,for,a,hike],[]), **infer(R,R1).**
  - R = [t<s,t=last_month(s),subset(e,t),t<s],
  - R1 = [e<s]

Exercise 1

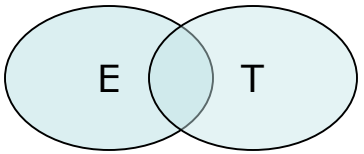
- Homework Question A (2pts)
  - Run
    - *Tomorrow, I will go for a hike*
  - Give the result
- Homework Question B (2pts)
  - What should I be able to infer?
- Homework Question C (4pts)
  - Add an inference rule to do this

Exercise 2

- Consider now
  - Yesterday, Noah had a rash
- Let
  - T = yesterday(S)
  - E = interval in which Noah is in a state of having a rash
  - T < S
  - E ∩ T ≠ ∅



?- sbar(R,[yesterday,noah,had,a,rash],[]).
R = [t<s,t=yesterday(s),intersect(e,t),t<s]



notation: define intersect(e,t) to mean E intersects T is non-empty

Exercise 2

- Homework Question (8pts)
  - Give a diagram explanation (see slide 10) of how
  - R = [t<s,t=yesterday(s),intersect(e,t),t<s]
  - is computed piece-by-piece
  - for the query
  - ?- sbar(R,[yesterday,noah,had,a,rash],[]).

Exercise 3

- **Theme**: *Inconsistency*
- Homework Question (8pts)
- Explain formally what is wrong with the following sentences:
  - (i) # Yesterday, I will go for a hike
  - (ii) # Tomorrow, Noah had a rash

  - # = semantically odd
- **hint**: Run the sentences...

Exercise 3

- **Extra Credit (10pts)**
- Write a Prolog rule
  - inconsistent(R)
  - that succeeds when it detects a logical inconsistency in the list of relations R
  - your rule should detect the inconsistency in sentences (i) and (ii)
  - hint: it's only one rule

```
 ?- sbar(R,[yesterday, i,will,go,for,a,hike],[]), inconsistent(R).
R = [t<s,t=yesterday(s),subset(e,t),s<t]
yes
| ?- sbar(R,[yesterday, i,will,go,for,a,hike],[]), \+ inconsistent(R).
no
```