T-(538|725)-MALV, Málvinnsla
Merkingarfræði

Hrafn Loftsson[1] Hannes Högni Vilhjálmsson[1]

[1]Tölvunarfræðideild, Háskólinn í Reykjavík

Október 2007

Outline

Outline

Merkingarfræði (e. semantics)

Greining merkingar (e. semantic analysis)

- Það að búa til einhverja formgerð (e. representation) fyrir setningu.
- Dæmi um formgerð er rökrænt form (e. logical form).
- Það form getur síðan t.d. verið notað til að ákvarða hvort tiltekin setning sé sönn eða ósönn.

Principle of compositionality

- Stundum hægt að búa til formgerðina um leið og setning er þáttuð.
- Byggir á "Principle of compositionality":
  - "it is possible to compose the meaning of a sentence from the meaning of its parts"

Merkingarfræði (e. semantics)

Greining merkingar (e. semantic analysis)

- Það að búa til einhverja formgerð (e. representation) fyrir setningu.
- Dæmi um formgerð er rökrænt form (e. logical form).
- Það form getur síðan t.d. verið notað til að ákvarða hvort tiltekin setning sé sönn eða ósönn.

Principle of compositionality

- Stundum hægt að búa til formgerðina um leið og setning er þáttuð.
- Byggir á "Principle of compositionality":
  - "it is possible to compose the meaning of a sentence from the meaning of its parts"

Outline

Lambda reikningur (e. λ calculus)

Hvað er?

- Fallareikningur – Church (1941)
- `http://en.wikipedia.org/wiki/Lambda_calculus`
- Getum notað Lambda reikning til að varpa setningaliðum yfir í λ-*segðir* (föll) (e. λ-*expressions*).

Dæmi

- "is a waiter" $= \lambda x.waiter(x)$ (kallað λ-abstraction)
- $\lambda x.waiter(x)(Bill) = waiter(Bill)$ (kallað β-reduction)

Lambda reikningur (e. λ calculus)

Hvað er?

- Fallareikningur – Church (1941)
- `http://en.wikipedia.org/wiki/Lambda_calculus`
- Getum notað Lambda reikning til að varpa setningaliðum yfir í λ-*segðir* (föll) (e. λ-*expressions*).

Dæmi

- "is a waiter" = $\lambda x.waiter(x)$ (kallað λ-abstraction)
- $\lambda x.waiter(x)(Bill) = waiter(Bill)$ (kallað β-reduction)

Lambda reikningur í Prolog

- Getum líkt eftir lambda reikningi í Prolog.
- Setjum λ-segðir inn í DCG reglurnar.
- Merkingarlega formgerðin (e. semantic representation) er þá búin til smám saman um leið og þáttað er.
- Algengt að láta `X^waiter(X)` standa fyrir λx.waiter(x)
- Skoðum nú Prolog kóða sem getur búið til formgerð fyrir setningar eins og *Mark is a waiter* (dæmi 1) ...
  - ..... en fyrst án nokkrar merkingargreiningar (dæmi 0)
- og *Mr. Schmidt called Bill* (dæmi 2)

Prolog dæmi 0

```
s --> np, vp.
vp --> verb, np.

np --> ['Bill'].
np --> ['Mark'].
np --> det, noun.

noun --> [waiter].
det --> [a].
verb --> [is].
```

- ?- s(['Mark', is, a, waiter], []).

Prolog dæmi 1

```
s(Predicate) --> np(Subject), vp(Subject^Predicate).
vp(Semantics) --> verb, np(Semantics).

np('Bill') --> ['Bill'].
np('Mark') --> ['Mark'].
np(X) --> det, noun(X).

noun(X^waiter(X)) --> [waiter].
det --> [a].
verb --> [is].
```

- ?- s(S, ['Mark', is, a, waiter], []).
- ?- s(waiter('Bill'), L, []).

Prolog dæmi 2

```
s(Semantics) --> np(Subject), vp(Subject^Semantics).

vp(Subject^Semantics) --> verb(Subject^Semantics).
vp(Subject^Semantics) --> verb(Object^Subject^Semantics),
                          np(Object).

np('Bill') --> ['Bill'].
np('Mr. Schmidt') --> ['Mr. Schmidt'].

verb(X^rushed(X)) --> [rushed].
verb(Y^X^called(X,Y)) --> [called].
```

- ?- s(S, ['Bill', rushed], []).
- ?- s(S, ['Mr. Schmidt', called, 'Bill'], []). (sjá tré bls. 206)

Outline

Umsagnarrökfræði/Umsagnarreikningur (e. (First Order) predicate calculus (FOPC))

- Þurfum að geta táknað "state of the world" á einhvern máta.
- Algengt að nota **umsagnar-röklíðaformgerðir** (e. predicate-argument structures)
- Varpar orðum, liðum og setningum yfir í tákn (e. symbol) og formgerðir sem standa fyrir hluti og eiginleika í tilteknu samhengi (e. **universe of discourse**).
- **Umsagnarrökfræði** er þægilegt tæki sem notað er í þessum tilgangi.
- `http://en.wikipedia.org/wiki/First-order_logic`
- Prolog er byggt á FOPC.

Umsagnarrökfræði

- Fullyrðingar eru settar fram sem **liðir** (e. terms)
- Einfaldir liðir:
  - **Fastar** (e. constants), t.d. 'Socrates', 'Pierre'
  - **Breytur** (e. variables), t.d. X, Y, Z.
- Samsettir liðir:
  - Standa fyrir eiginleika eða **vensl**, t.d.:
  - person('Pierre').
  - object(table).
  - on('Pierre', table).

Framsetning á nafnorðum og lýsingarorðum

- Orð eins og *waiter*, *patron*, *yellow*, *hot* hafa eiginleika sem við vörpum yfir í umsagnir (e. predicates) með einum röklið (e. argument).
- $\lambda x.waiter(x)$, í Prolog: `X^waiter(X)`
  - $\lambda x.waiter(x)(Bill) = waiter(Bill)$
- $\lambda x.hot(x)\ meal(x)$, í Prolog: `X^(hot(X), meal(X))`

Framsetning á sögnum og forsetningum

- Sagnir eins og *run*, *bring* og *serve* eru vensl sem við vörpum yfir í umsagnir með 1-2 rökliði eftir því hvort um sjálfstæðar eða ósjálfstæðar sagnir er að ræða.
- $\lambda x.ran(x)$, í Prolog: `X^ran(X)`
  - $\lambda x.ran(x)(Pierre) = ran(Pierre)$
- $\lambda y \lambda x.brought(x,y)$, í Prolog: `Y^X^brought(X,Y)`
  - *Roby brought a plate*: `brought('Roby',Z^plate(Z))`
- Forsetningar tengja oft saman tvo nafnliði:
  - `Y^X^with(X, Y)`, *The table with a napkin*, with(`Z^table(Z)`, `T^napkin(T)`)

Magnarar (e. quantifiers)

1. *A waiter ran.*
2. *Every waiter ran.*
3. *The waiter ran.*

Þessar setningar hafa mjög mismunandi merkingu þó munurinn á þeim felist eingöngu í ákvæðisorðinu (e. determiner)

Magnarar

- **existential quantifier**, $\exists$.
  - $\exists$x.P, til er x þannig að P sé sönn.
- **universal quantifier**, $\forall$.
  - $\forall$x.P, Fyrir öll x, P er sönn.
- **restricted existential quantifier**, $\exists!$.
  - $\exists!$x.P, til er nákvæmlega eitt x þannig að P sé sönn.

Magnarar

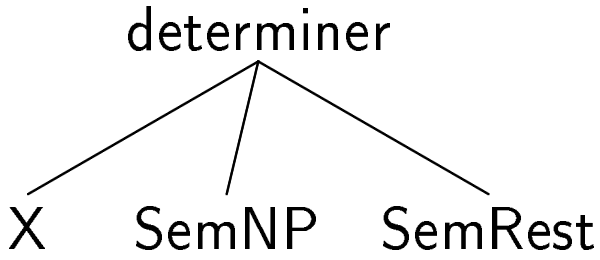
Setning	**Rökrænt form**
A waiter ran	$\exists x(\text{waiter}(x) \wedge \text{ran}(x))$ exists(X, waiter(X), ran(X))
Every waiter ran	$\forall x(\text{waiter}(x) \Rightarrow \text{ran}(x))$ all(X, waiter(X), ran(X))
The waiter ran	$\exists !x(\text{waiter}(x) \wedge \text{ran}(x))$ the(X, waiter(X), ran(X))

Magnarar

Setning	**Rökrænt form**
A waiter brought a meal	$\exists x(\text{waiter}(x) \wedge \exists y(\text{meal}(y) \wedge \text{brought}(x,y)))$ exists(X, waiter(X), exists(Y, meal(Y), brought(X,Y))
Every waiter brought a meal	$\forall x(\text{waiter}(x) \Rightarrow \exists y(\text{meal}(y) \wedge \text{brought}(x,y))))$ all(X, waiter(X), exists(Y, meal(Y), bruoght(X,Y)))
The waiter brought a meal	$\exists! x(\text{waiter}(x) \wedge \exists y(\text{meal}(y) \wedge \text{brought}(x,y)))$ the(X, waiter(X), exists(Y, meal(Y), brought(X,Y)))

Magnarar

- Lendum í vandræðum með orð eins og *two*, *three*, *several*, *many*, *this*, *that*, o.s.frv.
- Í stað þess að nota nöfn á mögnurum er hægt að nota ákvæðisorðin sjálf sem samsettan lið í Prolog:
- *A waiter brought a meal*
  - a(X, waiter(X), a(Y, meal(Y), brought(X,Y)))
- *Two waiters brought our meals*
  - two(X, waiter(X), our(Y, meal(Y), brought(X,Y)))



Prolog dæmi 3

```
s(Sem) --> np((X^SemRest)^Sem), vp(X^SemRest).

np((X^SemRest)^Sem) --> determiner((X^SemNP)^(X^SemRest)^Sem), noun(X^SemNP).

vp(X^SemRest) --> verb(X^SemRest).
vp(X^SemRest) --> verb(Y^X^SemVerb), np((Y^SemVerb)^SemRest).

noun(X^waiter(X)) --> [waiter].
noun(X^patron(X)) --> [patron].
noun(X^meal(X)) --> [meal].

verb(X^rushed(X)) --> [rushed].
verb(Y^X^ordered(X,Y)) --> [ordered].
verb(Y^X^brought(X,Y)) --> [brought].

determiner((X^SemNP)^(X^SemRest)^a(X, SemNP, SemRest)) --> [a].
determiner((X^SemNP)^(X^SemRest)^the(X, SemNP, SemRest)) --> [the].

?- s(Sem, [the, patron, ordered, a, meal], []).
Sem = the(_G549, patron(_G549), a(_G576, meal(_G576), ordered(_G549, _G576)))
Þ.e. Sem = the(X, patron(X), a(Y, meal(Y), ordered(X, Y)))
```