The State of Affairs

Two people at a table, Pierre and Socrates, and a robot waiter.
Formal Semantics

Its goal is to:

- Represent the state of affairs.
- Translate phrases or sentences such as *The robot brought the meal* or *the meal on the table* into logic formulas.
- Solve references: Link words to real entities.
- Reason about the world and the sentences.

A way to represent things and relations is to use first-order predicate calculus (FOPC) and predicate–argument structures.
Predicates

Constants:

% The people:
'Socrates'.
'Pierre'.

% The chairs:
chair1.  % chair #1
chair2.  % chair #2

% The unique table:
table1.  % table #1

Predicates to encode properties:

person('Pierre').
person('Socrates').

object(table1).
object(chair1).
object(chair2).

chair(chair1).
chair(chair2).
table(table1).

Predicates to encode relations:

in_front_of(chair1, table1).
on('Pierre', table1).
Prolog is a natural tool to do first-order predicate calculus

- Things, either real or abstract, are mapped onto constants — or atoms: 'Socrates', 'Pierre', chair1, chair2.
- Predicates can encode properties: person('Pierre'), person('Socrates'), object(table1), object(chair1).
- Predicates can encode relations: in_front_of(chair1, table1), on('Pierre', table1).
- Variables unify with objects
Querying the State of Affairs

Constants:
?- table(chair1).
false.
?- chair(chair2).
true.

Variables:
?- chair(X).
X = chair1;
X = chair2

Conjunctions:
?- chair(X), in_front_of(X, Y), table(Y).
X = chair1, Y = table1
Logical forms map sentences onto predicate-argument structures

*I would like to book a late flight to Boston*

\[
\text{would(like_to(i, book(i, np_pp(a(late(flight)), X^to(X, boston))))))}
\]
The principle of compositionality assumes that a sentence’s meaning depends on the meaning of its phrases. “The meaning of the whole is a function of the meaning of its parts.”

A complementary assumption is that phrases carrying meaning can be mapped onto constituents – syntactic units.

The principle of compositionality ties syntax and semantics together. We saw that a predicate-argument structure could represent a sentence – the whole. How to represent the parts – the constituents?
The λ-calculus is a device to abstract properties or relations.

\[ \lambda x. \text{property}(x) \]

or

\[ \lambda y. \lambda x. \text{relation}(x, y) \]

A λ-expression is incomplete until a value has been given to it. Supplying such a value is called a β-reduction.

\[ (\lambda x. \text{property}(x))\text{entity}\#1 \]

yields

\[ \text{property(entity}\#1) \]

In Prolog, \( X \sim \text{property}(X) \) represents \( \lambda x. \text{property}(x) \)
**Nouns**

Proper nouns: *Mark, Nathalie, Ludwig*

Common nouns (properties): *lecturer, book:*

\[
\lambda x.\text{lecturer}(x) \quad \lambda x.\text{lecturer}(x)(\text{Bill}) = \text{lecturer}(\text{Bill})
\]

Adjectives

\[
\lambda x.\text{big}(x) \quad \lambda x.\text{big}(x)(\text{Bill}) = \text{big}(\text{Bill})
\]

Adjectives and nouns: *big table*

\[
\lambda x.(\text{big}(x) \land \text{table}(x))
\]

Noun compounds are difficult: *lecture room*

\[
\lambda x.(\text{lecture}(x) \land \text{room}(x)) \quad ?? \text{Wrong!}
\]

A better form is:

\[
\lambda x.((\text{modify}(x, \text{lecture}) \land \text{room}(x))
\]

although not completely satisfying
Verbs of being are similar to adjectives or nouns

**Intransitive verbs**
\[ \lambda x.\text{rushed}(x) \]
\[ \lambda x.\text{rushed}(x)(\text{Bill}) = \text{rushed}(\text{Bill}) \]

**Transitive verbs**
\[ \lambda y.\lambda x.\text{ordered}(x, y) \]

**Prepositions**
\[ \lambda y.\lambda x.\text{to}(x, y) \]
Determiners

A caterpillar is eating
\[ \exists x, \text{caterpillar}(x) \land \text{eating}(x), \text{ or } \exists (X, \text{caterpillar}(X), \text{eating}(X)) \]

Every caterpillar is eating
\[ \forall x, \text{caterpillar}(x) \Rightarrow \text{eating}(x), \text{ or } \text{all}(X, \text{caterpillar}(X), \text{eating}(X)) \]

A caterpillar is eating a hedgehog
\[ \exists x, \text{caterpillar}(x) \land (\exists y, \text{hedgehog}(y) \land \text{eating}(x,y)), \text{ or } \exists (X, \text{caterpillar}(X), \exists (Y, \text{hedgehog}(Y), \text{eating}(X, Y)) \]

Every caterpillar is eating a hedgehog
\[ \forall x, \text{caterpillar}(x) \Rightarrow (\exists y, \text{hedgehog}(y) \land \text{eating}(x,y)), \text{ or } \text{all}(X, \text{caterpillar}(X), \text{exists}(Y, \text{hedgehog}(Y), \text{eating}(X, Y)) \]
Determiners: An Example

*Two waiters brought our meals*

translated into

two(X, waiter(X), our(Y, meal(Y), brought(X, Y)))
General Representation of Determiners

\[
\text{determiner} \quad \text{Semantics of the noun phrase: } \text{SemNP}(X)\ldots \quad \text{Semantics of the rest of the sentence: } \text{SemRest}
\]

Representation:

\[
(X^\text{NP})^\wedge (X^\text{Rest})^\wedge a(X, \text{NP}, \text{Rest})
\]
The partial, intermediate representations of

*The waiter brought the meal* are

\[
\begin{align*}
\text{waiter} & \quad X^\text{waiter}(X) \\
\text{The waiter} & \quad (X^\text{Rest})^\text{the}(X, \text{waiter}(X), \text{Rest}) \\
\text{brought} & \quad Y^X^\text{brought}(X, Y) \\
\text{meal} & \quad Y^\text{meal}(Y) \\
\text{the meal} & \quad (Y^\text{Verb})^\text{the}(Y, \text{meal}(Y), \text{Verb})
\end{align*}
\]

The operation to compose *brought the meal* is more complex. It should produce something like:

\[X^\text{the}(Y, \text{meal}(Y), \text{brought}(X, Y))\]
We parse the verb phrase *brought the meal* using the rule

\[ \text{vp(SemVP)} \rightarrow \text{verb(SemVerb)}, \text{np(SemNP)}. \]

We have:

\[ \text{SemVerb} = Y^X^\text{brought}(X, Y) \]
\[ \text{SemNP} = (Y^\text{Verb})^\text{the}(Y, \text{meal}(Y), \text{Verb}) \]

We just write the unification: \( \text{Verb}=\text{brought}(X, Y) \)

Prolog returns:

```prolog
?- \( \text{SemVerb} = Y^X^\text{brought}(X, Y), \)
\( \text{SemNP} = (Y^\text{Verb})^\text{the}(Y, \text{meal}(Y), \text{Verb}), \)
\( \text{Verb}=\text{brought}(X, Y). \)
\( \text{SemVerb} = Y^X^\text{brought}(X, Y), \)
\( \text{SemNP} = (Y^\text{brought}(X, Y))^\text{the}(Y, \text{meal}(Y), \text{brought}(X, Y)), \)
\( \text{Verb} = \text{brought}(X, Y). \)
```
Compositionality: The Lexicon

class noun {X^waiter(X)) { waiter.}
class noun {X^patron(X)) { patron.}
class noun {X^meal(X)) { meal.}
class verb {X^rushed(X)) { rushed.}
class verb {Y^X^ordered(X, Y)) { ordered.}
class verb {Y^X^brought(X, Y)) { brought.}
class determiner {((X^NP)^((X^Rest)^a(X, NP, Rest))) { a.}
class determiner {((X^NP)^((X^Rest)^the(X, NP, Rest))) { the.}
s(Semantics) --> np((X^Rest)^Semantics), vp(X^Rest).
np((X^Rest)^SemDet) -->
  determiner((X^NP)^((X^Rest)^SemDet)),
  noun(X^NP).
vp(Subject^Verb) --> verb(Subject^Verb).
v p(Subject^Predicate) -->
  verb(Object^Subject^Verb),
  np((Object^Verb)^Predicate).

?- s(Semantics, [the, patron, ordered, a, meal], []).
Semantics = the(_4, patron(_4), a(_32, meal(_32), ordered(_4, _32)))
A hedgehog has a nest

\[
a(X, \text{hedgehog}(X), a(Y, \text{nest}(Y), \text{have}(X, Y)).
\]

?- \text{hedgehog}(X), a(Y, \text{nest}(Y), \text{have}(X, Y)).

\[
\text{exists}(X, \text{Property1}, \text{Property2}) :-
\text{Property1},
\text{Property2},
!.
\]
Resolving References: all

*All hedgehogs have a nest*

all(X, hedgehog(X), a(Y, nest(Y), have(X, Y)).

*There is no hedgehog, which has no nest*

all(X, Property1, Property2) :-
  \+, (Property1, \+, Property2), Property1, !.
Application: Spoken Language Translator (Agnäs et al. 1994)

English     What is the earliest flight from Boston to Atlanta?
French      Quel est le premier vol Boston-Atlanta?

English     Show me the round trip tickets from Baltimore to Atlanta
French      Indiquez-moi les billets aller-retour Baltimore-Atlanta

English     I would like to go about nine am
French      Je voudrais aller aux environs de 9 heures

English     Show me the fares for Eastern Airlines flight one forty seven
French      Indiquez-moi les tarifs pour le vol Eastern Airlines cent quarante sept
Semantic Interpretation

Question:

What is the earliest flight from Boston to Atlanta?

Modeling a flight from Boston to Atlanta:

\[\exists x (\text{flight}(x) \land \text{from}(x, \text{Boston}) \land \text{to}(x, \text{Atlanta}) \land \exists y (\text{time}(y) \land \text{departs}(x, y)))\]

Finding the earliest flight:

\[\arg\min_y \exists x (\text{flight}(x) \land \text{from}(x, \text{Boston}) \land \text{to}(x, \text{Atlanta}) \land \\
\exists y (\text{time}(y) \land \text{departs}(x, y)))\]

SLT uses the logical form as a universal representation, independent from the language.

It converts sentences from and to this representation.
Semantic Parsing

SLT does not use variables for the nouns.

*I would like to book a late flight to Boston*

is converted into the Prolog term:

\[
\text{would(like_to(i, book(i, np_pp(a(late(flight)), X^to(X, boston)))))}
\]
1 rule(s_np_vp,
    s([sem=VP]),
    [np([sem=NP,agr=Ag]),
     vp([sem=VP,subjsem=NP,aspect=fin,agr=Ag])]).

2 rule(vp_v_np,
    vp([sem=V,subjsem=Subj,aspect=Asp,agr=Ag]),
    [v([sem=V,subjsem=Subj,aspect=Asp,agr=Ag, subcat=[np([sem=NP])])]),
    np([sem=NP,agr=_])).

3 rule(vp_v_vp,
    vp([sem=V,subjsem=Subj,aspect=Asp,agr=Ag]),
    [v([sem=V,subjsem=Subj,aspect=Asp,agr=Ag, subcat=[vp([sem=VP,subjsem=Subj])])]),
    vp([sem=VP,subjsem=Subj,aspect=ini,agr=])).
### Lexicon entries

<table>
<thead>
<tr>
<th>#</th>
<th>Lexicon entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>lex(boston,np([sem=boston,agr=(3-s)]))</code>.</td>
</tr>
<tr>
<td>2</td>
<td><code>lex(i,np([sem,agr=(1-s)]))</code>.</td>
</tr>
<tr>
<td>3</td>
<td><code>lex(flight,n([sem=flight,num=s]))</code>.</td>
</tr>
<tr>
<td>4</td>
<td><code>lex(late,adj([sem=late(NBAR),nbarsem=NBAR]))</code>).</td>
</tr>
<tr>
<td>5</td>
<td><code>lex(a,det([sem=a(NBAR),nbarsem=NBAR,num=s]))</code>).</td>
</tr>
<tr>
<td>6</td>
<td><code>lex(to,prep([sem=X^to(X,NP),npsem=NP]))</code>).</td>
</tr>
<tr>
<td>7</td>
<td><code>lex(to,inf([]))</code>.</td>
</tr>
<tr>
<td>8</td>
<td><code>lex(book,v([sem=have(Subj,Obj),subjsem=Subj,aspect=ini,agr=_ ,subcat=[np([sem=Obj])])]))</code>.</td>
</tr>
<tr>
<td>9</td>
<td><code>lex(would,v([sem=would(VP),subjsem=Subj,aspect=fin,agr=_ ,subcat=[vp([sem=VP,aubjsem=Subj])])]))</code>.</td>
</tr>
<tr>
<td>10</td>
<td><code>lex(like,v([sem=like_to(Subj,VP),subjsem=Subj,aspect=ini,agr=_ ,subcat=[inf([])],vp([sem=VP,subjsem=Subj])])])</code>.</td>
</tr>
</tbody>
</table>
Transferring Logical Forms

\[ \text{trule}(\text{<Comment> } \text{<QLF pattern 1> } \text{<Operator> } \text{<QLF pattern 2>}) \].

Operator is \( \geq, =<, \) or \( \== \).

<table>
<thead>
<tr>
<th>Lexical rules</th>
<th>Syntactic rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{trule}([\text{eng, fre}], flight1 &gt; vol1).</td>
<td>\text{trule}([\text{eng, fre}], form(tr(relation,nn), tr(noun1), tr(noun2)) &gt;= [\text{and, tr(noun2), form(prep(tr(relation)), tr(noun1))}])</td>
</tr>
</tbody>
</table>
### Parallel Corpora (Swiss Federal Law)

<table>
<thead>
<tr>
<th>German</th>
<th>French</th>
<th>Italian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Art. 35 Milchtransport</strong></td>
<td><strong>Art. 35 Transport du lait</strong></td>
<td><strong>Art. 35 Trasporto del latte</strong></td>
</tr>
<tr>
<td>1 Die Milch ist schonend und hygienisch in den Verarbeitungsbetrieb zu transportieren. Das Transportfahrzeug ist stets sauber zu halten. Zusammen mit der Milch dürfen keine Tiere und milchfremde Gegenstände transportiert werden, welche die Qualität der Milch beeinträchtigen können.</td>
<td>1 Le lait doit être transporté jusqu’à l’entreprise de transformation avec ménagement et conformément aux normes d’hygiène. Le véhicule de transport doit être toujours propre. Il ne doit transporter avec le lait aucun animal ou objet susceptible d’en altérer la qualité.</td>
<td>1 Il latte va trasportato verso l’azienda di trasformazione in modo accurato e igienico. Il veicolo adibito al trasporto va mantenuto pulito. Con il latte non possono essere trasportati animali e oggetti estranei, che potrebbero pregiudicarne la qualità.</td>
</tr>
</tbody>
</table>

3 Wird Milch ausserhalb des Hofes zum Abtransport bereitgestellt, so ist sie zu beaufsichtigen.

3 Si le lait destiné à être transporté est déposé hors de la ferme, il doit être placé sous surveillance.

3 Se viene collocato fuori dall’azienda in vista del trasporto, il latte deve essere sorvegliato.
Alignment (Brown et al. 1993)

Canadian Hansard

```
And₁ the₂ program₃ has₄ been₅ implemented₆
```

```
Le₁ programme₂ a₃ été₄ mis₅ en₆ application₇
```

```
The₁ poor₂ don’t₃ have₄ any₅ money₆
```

```
Les₁ pauvres₂ sont₂ démunis₄
```

RDF: A popular graph format to encode knowledge.
SPARQL: A query language for RDF
In many ways, very similar to Prolog.

```
ilppp:Socrates rdf:type ilppp:person.
ilppp:table1 rdf:type ilppp:object.
ilppp:chair1 rdf:type ilppp:object.
ilppp:chair2 rdf:type ilppp:object.
```
RDF and SPARQL

Prolog:
?- object(X), object(Y), in_front_of(X, Y).
X = chair1,
Y = table1.

SPARQL:
SELECT ?x ?y
WHERE
{ ?x rdf:type ilppp:object.
  ?y rdf:type ilppp:object.
  ?x ilppp:in_front_of ?y
}

<table>
<thead>
<tr>
<th>Variables</th>
<th>?x</th>
<th>?y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>ilppp:chair1</td>
<td>ilppp:table1</td>
</tr>
</tbody>
</table>
DBpedia, Yago, Wikidata, and Freebase

Graph databases consisting of billions of RDF triples. Coming from a variety of sources such as Wikipedia infoboxes:

```{Infobox settlement
| name             | = Busan 
... 
| area_total_km2   | = 767.35 
... 
| population_total | = 3,614,950 
... 
}

DBpedia: The result of a systematic triple extraction from infoboxes

dbpedia:Busan foaf:name "Busan, Korea"@en .
dbpedia:Busan dbpedia-owl:populationTotal "3614950".
dbpedia:Busan dbpedia-owl:areaTotal "7.6735E8" .
...
SPARQL Endpoint

Network service accepting SPARQL queries such as:

```
SELECT ?entity ?population
WHERE
{
  ?entity foaf:name "Busan, Korea"@en.
  ?entity dbpedia-owl:populationTotal ?population.
}
```

that returns:

<table>
<thead>
<tr>
<th>Variables</th>
<th>entity</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td><a href="http://dbpedia.org/resource/Busan">http://dbpedia.org/resource/Busan</a></td>
<td>3614950</td>
</tr>
</tbody>
</table>

where http://dbpedia.org/resource/Busan or dbpedia:Busan is a unique entity name based on the Wikipedia web addresses (URI nomenclature).