Logic Programming
Applications

Michael Genesereth
Computer Science Department
Stanford University
Focus - Metaknowledge

Examples
  Dynamic Systems
  Boolean Logic
  Natural Language Processing
Example - Blocks World
Blocks World
Symbols: $a, b, c, d, e$

Unary Predicates:
  - $\text{clear}$ - blocks with no blocks on top.
  - $\text{table}$ - blocks on the table.

Binary Predicates:
  - $\text{on}$ - pairs of blocks in which first is on the second.
clear(a)
clear(d)
table(c)
table(e)
on(a,b)
on(b,c)
on(d,e)
External Actions

$u(a,b)$

$u(d,e)$
Symbols: \(a, b, c, d, e, s\)

Unary Constructors:
- `clear` - blocks with no blocks on top
- `table` - blocks on the table

Binary Constructors:
- `on` - pairs of blocks in which first is on the second
- `u(x, y)` - action of moving \(x\) from \(y\) to the table
- `s(x, y)` - action of moving \(x\) from the table to \(y\)
- `do(x, s)` - state resulting from action \(x\) in state \(s\)

Unary Predicate:
- `true(p, s)` - means that \(p\) is true in state \(s\)
true(on(X,Y),do(s(X,Y),S)) :-  
  true(clear(X),S) &  
  true(table(X),S) &  
  true(clear(Y),S)

true(table(X),do(u(X,Y),S)) :-  
  true(on(X,Y),S) &  
  true(clear(X),S)

true(clear(Y),do(u(X,Y),S)) :-  
  true(on(X,Y),S) &  
  true(clear(X),S)
true(\text{clear}(X), \text{do}(u(U,V),S)) :-
\hspace{1em} true(\text{clear}(X),S)

true(\text{table}(X), \text{do}(u(U,V),S)) :-
\hspace{1em} true(\text{table}(X),S)

true(\text{on}(X,Y), \text{do}(u(U,V),S)) :-
\hspace{1em} true(\text{on}(X,Y),S) \& \text{distinct}(X,U)

true(\text{on}(X,Y), \text{do}(s(U,V),S)) :-
\hspace{1em} true(\text{on}(X,Y),S)

true(\text{clear}(Y), \text{do}(s(U,V),S)) :-
\hspace{1em} true(\text{clear}(Y),S) \& \text{distinct}(Y,V)

true(\text{table}(X), \text{do}(s(U,V),S)) :-
\hspace{1em} true(\text{table}(X),S) \& \text{distinct}(X,U)
External Actions

$u(a,b)$

$u(d,e)$
Given the following (e.g. `true(clear(a), s)`):

- clear(a)
- on(a, b)
- table(c)
- clear(d)
- on(b, c)
- table(e)
- on(d, e)

Operation:

- u(a, b)

Conclude (e.g. `true(clear(a), do(u(a, b), s))`):

- clear(a)
- on(b, c)
- table(a)
- clear(b)
- on(d, e)
- table(c)
- clear(d)
- table(e)

_N.B.: can be used to generate plans as well as simulate._
Example - Natural Language Processing
Good Sentences:
  Mary likes Pat.
  Mary likes Pat and Quincy.
  Pat and Quincy like Mary.

Bad Sentences:
  Mary Pat likes.
  Likes and Mary Pat Quincy.
<sentence> ::= <np> <vp>
<np> ::= <noun>
<np> ::= <noun> "and" <noun>
<vp> ::= <verb> <np>
<noun> ::= "mary" | "pat" | "quincy"
<verb> ::= "like" | "likes"
sentence(Z) :- np(X) & vp(Y) & append(X,Y,Z)
sentence(Z) :- np(X) & vp(Y) & append(X,Y,Z)

np(X) :- noun(x)
np(Z) :- noun(X) & noun(Y) & append(X,and,Y,Z)
sentence(Z) :- np(X) & vp(Y) & append(X, Y, Z)

np(X) :- noun(X)
np(Z) :- noun(X) & noun(Y) & append(X, and, Y, Z)

vp(Z) :-
    verb(X) & np(Y) & append(X, Y, Z)
sentence(Z) :- np(X) & vp(Y) & append(X,Y,Z)

np(X) :- noun(x)
np(Z) :- noun(X) & noun(Y) & append(X,and,Y,Z)

vp(Z) :- verb(X) & np(Y) & append(X,Y,Z)

noun(mary)
noun(pat)
noun(quincy)
Logical Grammar

sentence(Z) :- np(X) & vp(Y) & append(X,Y,Z)

np(X) :- noun(x)
np(Z) :- noun(X) & noun(Y) & append(X,and,Y,Z)

vp(Z) :- verb(X) & np(Y) & append(X,Y,Z)

classical: noun(mary)
classical: noun(pat)
classical: noun(quincy)

classical: verb(like)
classical: verb(likes)
Examples

Sentences:
√ Mary likes Pat.
√ Mary likes Pat and Quincy.
√ Pat and Quincy like Mary.

Not Sentences:
× Mary Pat likes.
× Likes and Mary Pat Quincy.
Sentences:
   Mary likes Pat.
   Mary likes Pat and Quincy.
   Pat and Quincy like Mary.

Allowed but not sentences in natural English:
   Mary like Pat.
   Pat and Quincy likes Mary.

How can we enforce subject-verb number agreement?
Augmented Logical Grammar

sentence(Z) :- np(X,W) & vp(Y,W) & append(X,Y,Z)

np(X,0) :- noun(x)
np(Z,1) :- noun(X) & noun(Y) & append(X,and,Y,Z)

vp(Z,W) :- verb(X,W) & np(Y) & append(X,Y,Z)

noun(mary)
noun(pat)
noun(quincy)

verb(like,1)
verb(likes,0)
Augmented Logical Grammar

sentence($Z$) :- np($X$,$W$) & vp($Y$,$W$) & append($X$,$Y$,$Z$)

np($X$,0) :- noun($x$)
np($Z$,1) :- noun($X$) & noun($Y$) & append($X$,and,$Y$,$Z$)

vp($Z$,$W$) :- verb($X$,$W$) & np($Y$) & append($X$,$Y$,$Z$)

noun(mary)
noun(pat)
noun(quincy)

verb(like,1)
verb(likes,0)
sentence(Z) :- np(X,W) & vp(Y,W) & append(X,Y,Z)

np(X,0) :- noun(x)
np(Z,1) :- noun(X) & noun(Y) & append(X,and,Y,Z)

vp(Z,W) :- verb(X,W) & np(Y) & append(X,Y,Z)

noun(mary)
noun(pat)
noun(quincy)

verb(like,1)
verb(likes,0)
Augmented Logical Grammar

sentence(Z) :- np(X,W) & vp(Y,W) & append(X,Y,Z)

np(X,0) :- noun(x)
np(Z,1) :- noun(X) & noun(Y) & append(X,and,Y,Z)

vp(Z,W) :- verb(X,W) & np(Y) & append(X,Y,Z)

noun(mary)
noun(pat)
noun(quency)

verb(like,1)
verb(likes,0)
Example - Boolean Logic
Logical Expressions:
\(~p \& q \Rightarrow q \mid r\)

Proofs:
Given \((p \Rightarrow q)\) and \((q \Rightarrow r)\), prove that \((p \Rightarrow r)\).
Basic idea: represent expressions in Boolean Logic as Logic Programming terms and write rules to define basic concepts of Boolean Logic.

NB: We can extend to defining Logic Programming within Logic Programming as well. The formalization is messier, and some nasty problems need to be handled (notably paradoxes).
Symbols (propositions)

\[ p \quad q \quad r \]

Constructors

\[ \text{not}(p) \quad \text{if}(p,q) \]
\[ \text{and}(p,q) \quad \text{iff}(p,q) \]
\[ \text{or}(p,q) \]

Predicates

\[ \text{proposition}(p) \quad \text{implication}(x) \]
\[ \text{negation}(x) \quad \text{biconditional}(x) \]
\[ \text{conjunction}(x) \quad \text{sentence}(x) \]
\[ \text{disjunction}(x) \]
Syntactic Metadefinitions

proposition(p)
proposition(q)
proposition(r)

negation(not(X)) :- sentence(X)
conjunction(and(X,Y)) :- sentence(x) & sentence(y)
disjunction(or(x,y)) :- sentence(x) & sentence(y)
implication(if(x,y)) :- sentence(x) & sentence(y)
biconditional(iff(x,y)) :- sentence(x) & sentence(y)

sentence(x) :- proposition(x)
sentence(x) :- negation(x)
sentence(x) :- conjunction(x)
sentence(x) :- disjunction(x)
sentence(x) :- implication(x)
sentence(x) :- biconditional(x)
And Introduction:

\[ ai(X,Y,\text{and}(X,Y)) : - \\
    \text{sentence}(X) \ & \ \text{sentence}(Y) \]

Example:

\[ ai(p,q,\text{and}(p,q)) \]
\[ ai(\text{if}(p,q),\text{if}(q,r),\text{and}(\text{if}(p,q),\text{if}(q,r))) \]
And Elimination:

\[ ai(\text{and}(X,Y),X) :- \]
\[ \text{sentence}(X) \& \text{sentence}(Y) \]

\[ ai(\text{and}(X,Y),Y) :- \]
\[ \text{sentence}(X) \& \text{sentence}(Y) \]

Example:

\[ ai(\text{and}(p,q),p) \]
\[ ai(\text{and}(p,q),q) \]
\[ ai(\text{and}(\text{if}(p,q),\text{if}(q,r)),\text{if}(p,q)) \]
\[ ai(\text{and}(\text{if}(p,q),\text{if}(q,r)),\text{if}(q,r)) \]
Rule of Inference

Modus Ponens:

\[
mp(P, \text{if}(P, Q), Q) :- \\
\text{sentence}(P) \& \text{sentence}(Q)
\]

Example:

\[
mp(p, \text{if}(p, q), q) \\
mp(q, \text{if}(q, r), r) \\
mp(\text{if}(p, q), \text{if}(\text{if}(p, q), \text{if}(q, r)), \text{if}(q, r))
\]
Implication Introduction:

\[
\text{axiom}(\text{if}(P,\text{if}(Q,P))) \ :- \ \text{sent}(P) \ & \ \text{sent}(Q)
\]

e.g. \( p \Rightarrow (q \Rightarrow p) \)

Implication Distribution:

\[
\text{axiom}(\text{if}(\text{if}(P,\text{if}(Q,R)),\text{if}(\text{if}(P,Q),\text{if}(P,R))))) \\
\ :- \ \text{sent}(P) \ & \ \text{sent}(Q) \ & \ \text{sent}(R)
\]

e.g. \((p \Rightarrow (q \Rightarrow p)) \Rightarrow ((p \Rightarrow q) \Rightarrow p \Rightarrow r))\)
A sentence is *provable* from a set of premises if and only if it is a premise, an axiom, or the result of applying a rule of inference to provable sentences.

```
provable(Q) :- premise(Q)
provable(Q) :- axiom(Q)
provable(R) :- mp(P,Q,R) & provable(P)
provable(R) :- ai(P,Q,R) & provable(P)
provable(R) :- ae(P,Q) & provable(P)
```
Example

Premises:
if(p,q)
if(q,r)

Conclusion:
if(p,r)

Provability:
provable(if(p,q))
provable(if(q,r))
provable(if(if(q,r),if(p,if(q,r))))
provable(if(p,if(q,r)))
provable(if(p,if(q,r)),
   if(if(p,q),if(p,r))))
provable(if(if(p,q),if(p,r)))
provable(if(p,r))