Representation & Logic

Cosmo, The Wonder Dog

Propositional Calculus

A reminder

What makes up propositional calculus?

- A set of symbols: P, Q, R, S, true, false
- Connectives: ^, v, ¬, →.
- Rules to combine symbols and connectives into **well-formed sentences.**
- Semantics: an *interpretation*, or mapping from symbols into {T,F} – an assertion about their truth in some *possible world*.

Truth Tables

A quick way to understand the nature of a function, or to see if two functions are equivalent (have the same truth values). Show every possible combination of the variables. Here is a truth table for XOR (the exclusive OR)*:

x	y	$(x \vee y)$	$(x \wedge y)$	$\neg (x \wedge y)$	$(x \vee y) \wedge \neg (x \wedge y)$ *(DEF'N OF XOR)*
FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	TRUE	FALSE	TRUE	TRUE
TRUE	TRUE	TRUE	TRUE	FALSE	FALSE

Exercise

- Express “If Cosmo has cheese then he is happy.” in propositional calculus.
- Show that it is equivalent to “Either Cosmo doesn’t have cheese, or he is happy”, using a truth table.

Answer

- x = “Cosmo has cheese”.
- y = “Cosmo is happy”.

So, “If Cosmo has cheese then he is happy” is expressed as:

$x \rightarrow y$.

“Either Cosmo doesn’t have cheese, or he is happy” is expressed as:

$\neg x \vee y$.

Are they equivalent?

x	y	x → y
F	F	T
T	F	F
F	T	T
T	T	T

x	y	¬x	¬x v y
F	F	T	T
T	F	F	F
F	T	T	T
T	T	F	T

Predicate calculus

A very brief overview

What is predicate calculus?

- One way of representing knowledge.
- A formal reasoning system.

Why not just stick with propositional?

- Propositions are atomic – they can't be broken down. So, there is no relation whatsoever between P, "Cosmo likes cheese", and Q, "Cosmo likes cats".
- We would like to be able to reason over the parts of a proposition, e.g. "Cosmo", "likes" and "cats".

Predicate calculus syntax: symbols

Symbols start with a letter, and can contain letters, numbers and the underscore.

- *Constants (begin lowercase):* **george**, **tall**, **tree**, etc.
- *Truth symbols:* **true** and **false**.
- *Variables (begin uppercase):* **Last_Name**, **TOTAL**
- *Term:* constant, variable, or function expression.

Predicate calculus syntax: predicates

- *Predicate*: A function which names a relationship between zero or more objects.
- Given a semantics (later), predicates map to the range of truth values (i.e. **T** and **F**).
- For example, father(kim, john) maps to **T** under a certain interpretation.

Predicate calculus syntax: functions

- *Function expression:* **father(david)**, **plus(2,3)**
- Maps from elements in the domain to elements in the range. E.g. plus**(2,3)** maps to 5.
- *Arity* of a function: the number of elements in the *domain* mapped onto each element in the *range*. E.g. The arity of **plus(X,Y)** is 2.
- *Value:* the object in the range mapped onto. E.g. the value of **plus(5,6)** is 11.
- When talking about a function, we use its name and its arity, e.g. **plus/2**

Predicate calculus syntax: sentences

- *Atomic sentence*: a predicate, followed by a full stop. Examples: **friends(bill, george)**. **likes(father(david), bill). true. false.** *Also called propositions (recall prop. calc.)*
- Combine sentences with *logical connectives* (^**, v,** ¬**,** →**)** *universal* and *existential quantifiers (*∀**,** ∃), to make more complex sentences.
- All variables in sentences must be quantified. (e.g. **∀X(dog(X) →likes(X,bones).**)

Predicate calculus sentences

- Every atomic sentence is a sentence.
- If **s** and **t** are sentences, and **X** is a variable, then the following are sentences:
  - **¬s.** (not s)
  - **s^t.** (s and t)
  - **svt.** (s or t)
  - **s→t.** (s implies t)
  - **∀Xs.** (for all X, s)
  - **∃Xs.** (there exists an X such that s)

Semantics of PC

An *interpretation* over a *domain* D is an assignment of entities in D to constant, variable, predicate and function symbols, which is then used to determine the *truth value assignment* of sentences.

Symbol names might be evocative, but they have **no meaning** until an interpretation is provided.

e.g. father(kimbinsted,johnbinsted). Under what interpretation is this **T**? Can you think of an interpretation under which it would be **F**?

For PC expression X and interpretation I:

- If X has value T under I, then I *satisfies* X.
- If I satisfies X for all variable assignments, then I is a *model* of X.
- If a set of expressions is not satisfiable, it is *inconsistent*.
- If X has a value T for all possible interpretations, then X is *valid*.

First order predicate calculus

- First order PC allows quantified variables to refer to objects in the domain, but not to predicates.
- What kind of sentences might be better expressed in a higher-order PC?

Inference rules

- A PC expression **X** *logically follows* from a set of PC expressions **S** if every interpretation and variable assignment that satisfies **S** also satisfies **X**.
- An **inference rule** is *sound* if every PC expression produced by the rule from **S** also logically follows from **S**.
- An inference rule is *complete* if, given a set **S** of predicate calculus expressions, it can infer every expression that logically follows from **S**.

Some sound inference rules

Where P and Q are predicates:

- If **P** [is true] and **P ⇒ Q**, then **Q**.
- If **P ⇒Q** and ¬**Q**, then ¬ **P**.
- If **P** and **Q**, then **P^Q**.
- If **P^Q**, then both **P** and **Q**.

and so on… Notice that these are the same as the propositional calculus rules.

Unification

- Goal: determine whether or not two expressions match. Must find *variable bindings* that make expressions identical.
- Want the set of bindings to be as general as possible (most general unifier).
- See the function *unify* for a formal unification algorithm. (e.g. https://secure.wikimedia.org/wikipedia/en/wiki/Unification_%28computer_science%29)

Exercise

Unify the following, if possible. If not possible, explain why.

- loves(cosmo, cheese) & loves(X, Y).
- father(Y, kim) & father(son(peter), Z).
- foo(goo(X), X) & foo(Y, Y).

Answer

- loves(cosmo, cheese) & loves(X, Y).
  - X = cosmo
  - Y = cheese
- father(Y, kim) & father(son(peter), Z).
  - Y = son(peter)
  - Z = kim
- foo(goo(X), X) & foo(Y, Y).
  - Can’t be unified. Note that unification is a *syntactic* process.

Strengths of PC for knowledge representation

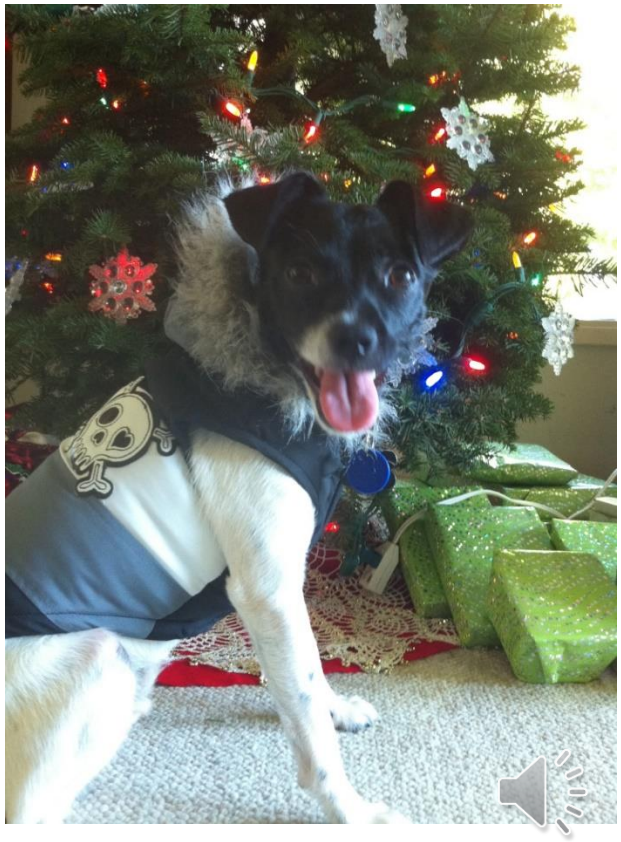
- Well-developed, well-understood formal system, with extensive inference rules.
- Highly expressive – that is, it can capture much of the knowledge we might want to represent.
- Given a clear semantics, allows strong, sound reasoning about the world.

Weaknesses of PC for knowledge representation

- First order PC does not allow meta-level reasoning (i.e. reasoning about reasoning).
- Only two truth values: no uncertainty, ambiguity, or fuzziness.
- *Requires* interpretation to have meaning – something that is often forgotten!

Write PC sentences for:

- Cosmo is a mutt and he loves cheese.
- If you give Cosmo cheese, he loves you.
- Cosmo is scared of everything.
- Some people are scared of Cosmo.
- Anyone who is scared of Cosmo is scared of everything.



Answers
(must also provide appropriate interpretation)

- Cosmo is a mutt and he loves cheese.
  - ∀X: mutt(cosmo) ^ (cheese(X) → ^ loves(cosmo,X)).
- If you give Cosmo cheese, he loves you.
  - ∀X ∀Y: cheese(X) ^ gives(Y, cosmo, X) → loves(cosmo, Y).
- Cosmo is scared of everything.
  - ∀X: scaredof(cosmo, X).
- Some people are scared of Cosmo.
  - ∃X: person(X) ^ scaredof(X, cosmo).
- Anyone who is scared of Cosmo is scared of everything.
  - ∀X ∀Y: scaredof(X,cosmo) → scaredof(X,Y).

Rough procedure

- What are the *specific* objects? These are constants (e.g. cosmo, kim, post_107, etc.).
- What are the classes/attributes of objects? These are 1 arity predicates (e.g. mutt/1, cheese/1, blue/1 etc.).
- What are the relationships/verbs? These are 1+ arity predicates (gives/3, likes/2, father/2 etc.).
- Is the sentence talking about *all* Xs (∀X), or just one or more of them (∃X)?
- Rewrite the sentence in pseudo-logic, and make sure the meaning is the same. Then, do the real thing!

More examples

- Mondays are horrible.
- If you drink coffee, Tuesdays are not horrible.
- Kim drinks coffee, so this Tuesday is not horrible.
- Cosmo doesn’t drink coffee.
- If Cosmo drinks coffee, his breath is horrible.

Fig 8.18 The blocks world.

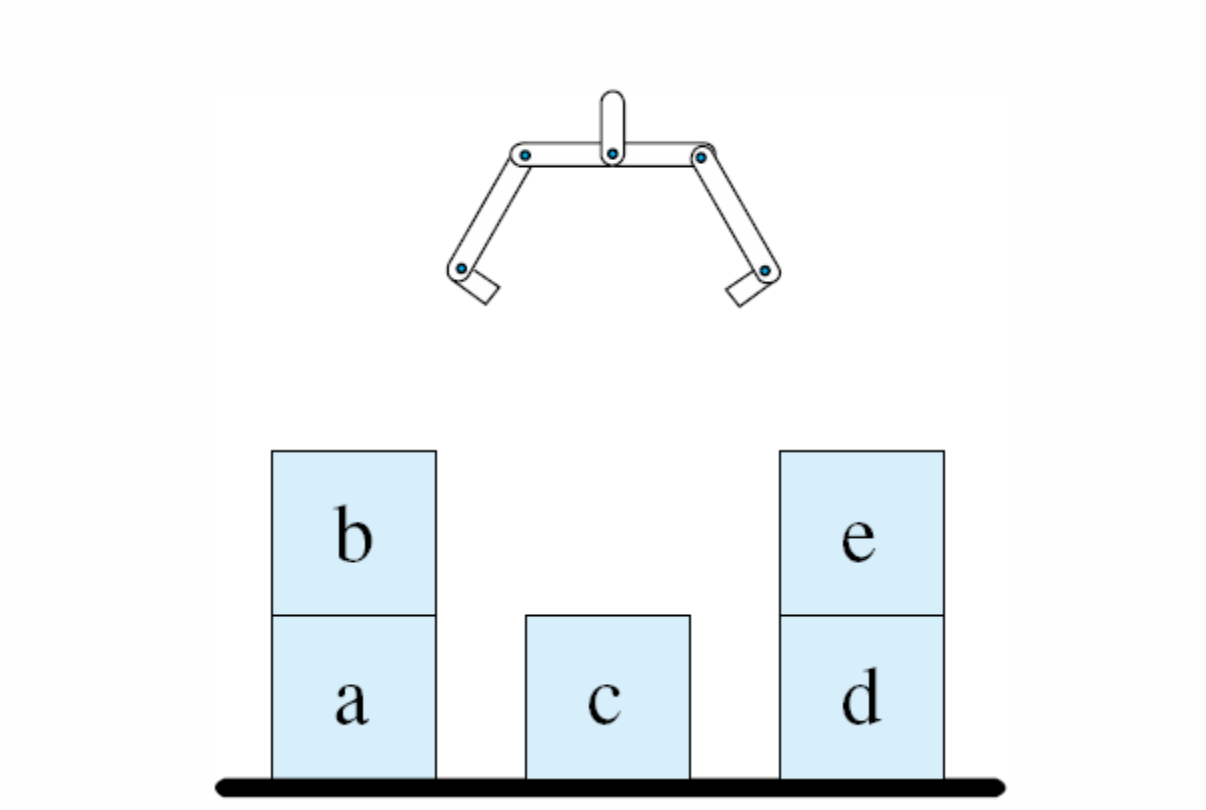
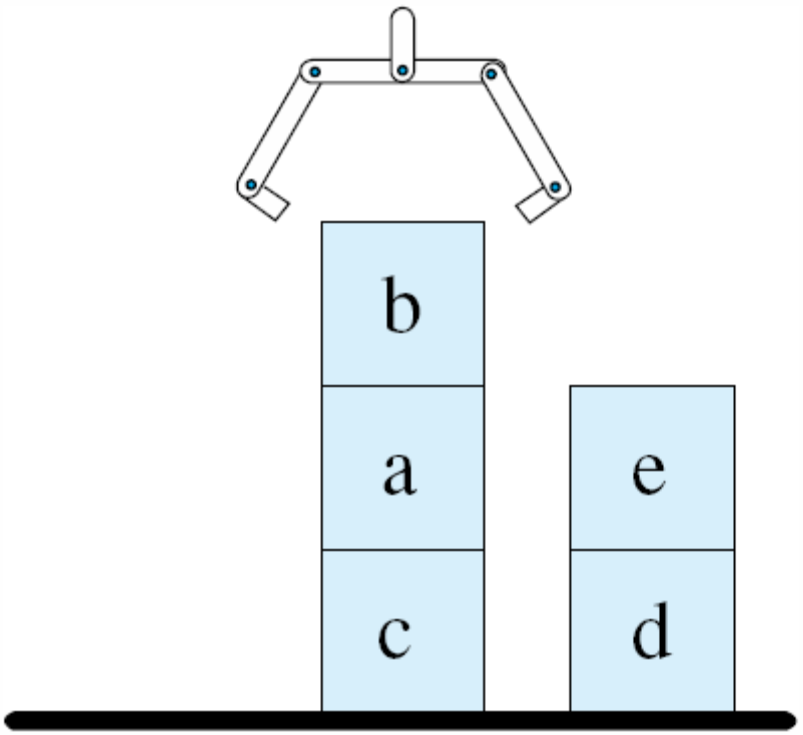


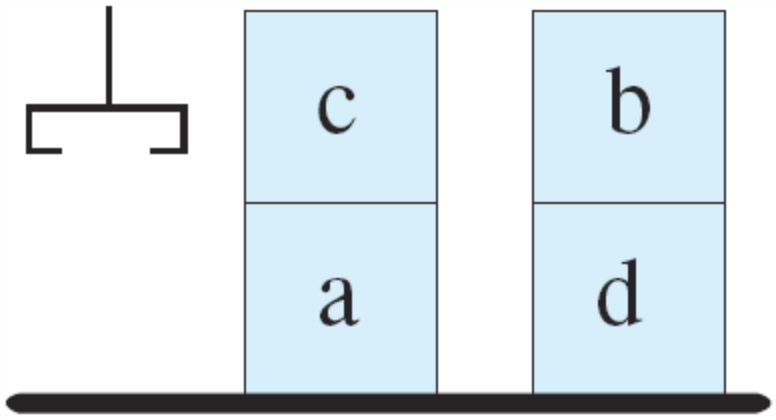
Fig 8.20 Goal state for the blocks world.



What predicates and constants would you use to describe Blocks World?

Remember, the goal is to describe it in such a way that you can reason about it.

Figure 2.3: A blocks world with its predicate calculate description.



on(c,a)
on(b,d)
ontable(a)
ontable(d)
clear(b)
clear(c)
hand_empty

The blocks world of figure 8.18 may now be represented by the following set of predicates.

STATE 1

ontable(a).	on(b, a).	clear(b).
ontable(c).	on(e, d).	clear(c).
ontable(d).	gripping( ).	clear(e).

STATE 2

ontable(a).	on(b,a).	clear(b).
ontable(c).	clear(c).	clear(d).
ontable(d).	gripping(e).	clear(e).

What are some useful truth relations? What about actions?

- Truth relations describe relationships between predicates. You can think of them as “in world” definitions of predicates.
- Actions have predicates as preconditions and consequences.

A number of truth relations or rules for performance are created for the clear (X), ontable (X), and gripping ().

1. $(\forall\ X)\ (\text{clear}(X) \leftarrow \neg\ (\exists\ Y)\ (\text{on}(Y,X)))$
2. $(\forall\ Y)\ (\forall\ X) \neg\ (\text{on}(Y,X) \leftarrow \text{ontable}(Y))$
3. $(\forall\ Y)\ \text{gripping}(\ ) \leftrightarrow \neg\ (\text{gripping}(Y))$
4. $(\forall\ X)\ (\text{pickup}(X) \rightarrow (\text{gripping}(X) \leftarrow (\text{gripping}(\ ) \wedge \text{clear}(X) \wedge \text{ontable}(X))))$.
5. $(\forall\ X)\ (\text{putdown}(X) \rightarrow ((\text{gripping}(\ ) \wedge \text{ontable}(X) \wedge \text{clear}(X)) \leftarrow \text{gripping}(X)))$.
6. $(\forall\ X)\ (\forall\ Y)\ (\text{stack}(X,Y) \rightarrow ((\text{on}(X,Y) \wedge \text{gripping}(\ ) \wedge \text{clear}(X)) \leftarrow (\text{clear}(Y) \wedge \text{gripping}(X))))$.
7. $(\forall\ X)(\forall\ Y)\ (\text{unstack}(X,Y) \rightarrow ((\text{clear}(Y) \wedge \text{gripping}(X)) \leftarrow (\text{on}(X,Y) \wedge \text{clear}(X) \wedge \text{gripping}(\ )))$.
8. $(\forall\ X)\ (\forall\ Y)\ (\forall\ Z)\ (\text{unstack}(Y,Z) \rightarrow (\text{ontable}(X) \leftarrow \text{ontable}(X)))$.
9. $(\forall\ X)\ (\forall\ Y)\ (\forall\ Z)\ (\text{stack}(Y,Z) \rightarrow (\text{ontable}(X) \leftarrow \text{ontable}(X)))$.

Using blocks example, the four operators pickup, putdown, stack, and unstack are represented as triples of descriptions (precondition, add, delete).

pickup(X)	P: gripping( ) $\wedge$ clear(X) $\wedge$ ontable(X) A: gripping(X) D: ontable(X) $\wedge$ gripping( )
putdown(X)	P: gripping(X) A: ontable(X) $\wedge$ gripping( ) $\wedge$ clear(X) D: gripping(X)
stack(X,Y)	P: clear(Y) $\wedge$ gripping(X) A: on(X,Y) $\wedge$ gripping( ) $\wedge$ clear(X) D: clear(Y) $\wedge$ gripping(X)
unstack(X,Y)	P: clear(X) $\wedge$ gripping( ) $\wedge$ on(X,Y) A: gripping(X) $\wedge$ clear(Y) D: gripping( ) $\wedge$ on(X,Y)

Fig 8.19 Portion of the state space for a portion of the blocks world.