Computational Semantics with Haskell

Yulia Zinova

Winter 2016/2017

We follow **?**, electronic access from the library

Semantics for Sea Battle

- Two steps:
  1. What is the reality the game is about?
  2. How to describe the way in which expressions (which are part of the game reality) relate to this reality?
- Reality of Sea Battle: a set of *states* of the game board
- A game state is a board with positions of ships indicated on it, and marks indicating the fields on the board that were under attack so far in the game.

Semantics for Sea Battle

- What is the meaning of the attack?

Semantics for Sea Battle

- What is the meaning of the attack?
- State change

Semantics for Sea Battle

- What is the meaning of the attack?
- State change
- Code: `http://www.computational-semantics.eu/FSemF.hs`
- A grid is a list of coordinates
- A game state is a number of grids, for convenience, we use a separate grid for each ship
- We need to check that ships do not overlap (in the code) and that each ship occupies adjacent squares (exercise)

See Battle: reactions

- Four reactions: *missed, hit, sunk, defeated*.
- Given a state and the position of the last attack, any of these reactions gives the value *true* or *false*. It can be expressed as a function from the set of states, the set of positions, and the set of reactions to *true/false*: F from $SxPxR$ to {0,1}
- Exercise: describe reactions in this terms

See Battle: reactions

- Four reactions: *missed, hit, sunk, defeated*.
- Given a state and the position of the last attack, any of these reactions gives the value *true* or *false*. It can be expressed as a function from the set of states, the set of positions, and the set of reactions to *true/false*: *F* from *SxPxR* to {0,1}
- Exercise: describe reactions in this terms
- Let us look at the implementation

See Battle: reactions

- Four reactions: *missed, hit, sunk, defeated*.
- Given a state and the position of the last attack, any of these reactions gives the value *true* or *false*. It can be expressed as a function from the set of states, the set of positions, and the set of reactions to *true/false*: F from $SxPxR$ to {0,1}
- Exercise: describe reactions in this terms
- Let us look at the implementation
- Exercise: implement *sunk* reaction

Sea Battle: pragmatics

- Gricean Maxims: cooperation, quality, quantity, mode of expression (?)
- **Cooperation**: Try to adjust every contribution to the spoken communication to what is percieved as the common goal of the communication at that point of interaction
- **Quality**: Aspire to truthfulness. Do not say anything you do not believe to be true. Do not say anything to which you have inadequate support.
- **Quantity**: Be as explicit as the situation requires, no more, no less.
- **Mode of expression**: Don't obscure, don't be ambiguous, aspire to conciseness and clarity.
- Why is the reaction *sunk* (when it is true) more informative, than *hit*?

Semantics of Propositional Logic

- What are the extralinguistic structures that propositional logic formulas are about?

Semantics of Propositional Logic

- What are the extralinguistic structures that propositional logic formulas are about?
- Pieces of information about the truth or falsity of atomic propositions
- This is encoded in *valuations*, functions from the set P of proposition letters to the set $\{0, 1\}$ of *truth values*.
- If V is such a valuation, then V can be extended to a function V^+ from the set of all propositional formulas to the set $\{0, 1\}$.
  - $V^+(p) = V(p)$ for all $p \in P$,
  - $V^+(\neg F) = 1$ iff all $V^+(F) = 0$,
  - $V^+(F_1 \wedge F_2) = 1$ iff $V^+(F_1) = V^+(F_2) = 1$,
  - $V^+(F_1 \vee F_2) = 1$ iff $V^+(F_1) = 1$ or $V^+(F_2) = 1$

Semantics of Propositional Logic: Definitions

- Formulas F for which the V^+ value does not depend on the V value of F are called *tautologies* ($\models F$)
- Formulas F with the property that $V^+(F) = 0$ for any V are called *contradictions*
- A formula F is *satisfiable* if there is at least one valuation V with $V^+(F) = 1$.
- A formula is called *contingent* is it is satisfiable but not a tautology.
- Two formulas F_1 and F_2 are called *logically equivalent* ($F_1 \equiv F_2$) if $V^+(F_1) = V^+(F_2)$ for any V
- Formulas $P_1 \ldots P_n$ (premises) logically imply ($P_1 \ldots P_n \models C$) formula C (conclusion) of every valuation which makes every member of $P_1 \ldots P_n$ true also makes C true

Propositional reasoning in Haskell

- propNames function
- A valuation for a propositional formula is a map from its variable names to the Booleans
- Function genVals generates the list of all valuations over a set of names
- Function allVals outputs a list of all valuations for a formula

Exercises

- Implement implication for a formula with a list of premises
- Implement a function that checks whether two propositional formulas are equivalent
- Reimplement the semantics of propositional formulas, using [String] type instead of [(String, Bool)] and indicating truth or falsity with presence/absence.

Semantics of Mastermind

- Five colours (red, yellow, blue, green, orange) and four positions
- How many settings are there? (with/without colour repetition)

Semantics of Mastermind

- Five colours (red, yellow, blue, green, orange) and four positions
- How many settings are there? (with/without colour repetition)
- Propositional logic: r_1 for ‘red occurs at position one’
- **Exercise:** find a formula that expresses that all positions have the same colour

Semantics of Mastermind: Implementation

- Assume the initial setting is *red, yellow, blue, blue*
- The game is won when

Semantics of Mastermind: Implementation

- Assume the initial setting is *red, yellow, blue, blue*
- The game is won when the correct formula $r_1 \wedge y_2 \wedge b_3 \wedge b_4$ is logically implied by the formulas that encode information about the rules of the fame and the information provided by the answers to guesses.
- Implementation: key element is the computation that determines appropriate reaction to a guess.
- To compute the reaction, first two kinds of elements need to be counted: elements that occur at the same position and elements that occur somewhere.
- We also need to update the list f possible patterns, given a particuar reaction, keeping all patterns that generate the same reaction and discard the rest.

Semantics of Mastermind: Exercise

- Modify the game function so that it can recognize stupid guesses (guesses that contradict information that was already supplied) and let the user know about them.

Semantics of predicate logic

- We will use three predicate letters: P, R, S: P is a one-place predicate, R is a two-place predicate, S is a three-place predicate.
- Extralinguistic structure should contain a domain of discourse D consisting of individual entities with an interpretation (I) for P, for R and for S.
- $I(P) \subseteq D$, $I(R) \subseteq D \times D$, $I(S) \subseteq D \times D \times D$
- A set of relation symbols (plus arities) specifies a predicate logical language L.
- A structure $M = (D, I)$ consisting of a non-empty domain D and an interpretation function I is called a *model for L*.

Inference Engine

- Natural language inference engine
- Aristotelian quantifiers
- Inferential pattern, syllogism
- Square of Opposition (`https://plato.stanford.edu/entries/square/image-a.jpg`): contraries, subcontraries
- Existential import: *No A are b* is taken to imply that there are *A*

Inference Engine: Knowledge Base

- Knowledge base: two relations (inclusion and non-inclusion)
- *All A are B*: $A \subseteq B$
- *No A are B*: $A \subseteq \bar{B}$
- *Some A are not B*: $A \not\subseteq B$
- *Some A are B*: $A \not\subseteq \bar{B}$
- A knowledge base is a list of triples (Class_1, Class_2, Boolean)
- (A, B, **True**) expresses $A \subseteq B$
- (A, B, **False**) expresses $A \not\subseteq B$

Relations: properties

- Relation R is transitive if...
- The *transitive closure* of R is the smallest transitive relation S with $R \subseteq S$
- Relation R is transitive if...
- The *reflexive transitive closure* of R is the smallest reflexive and transitive relation S with $R \subseteq S$
- How to compute the reflexive transitive closure?

References:

Grice, H. P. (1975). Logic and conversation. In P. Cole and J. L. Morgan, editors, *Syntax and Semantics 3: Speech acts*, pages 41–58. Academic Press, New York.

Van Eijck, J. and Unger, C. (2010). *Computational semantics with functional programming*. Cambridge University Press.