

February 14, 2006

CLASS 6: EVENTS, SETS & FUNCTIONS

OVERVIEW

Upshot of today's class:

- introduction to sub-atomic semantics
- sets and functions (lambda calculus)

EVENTS

(1) Shelby saw Hannibal.

- not only a relation between a seer and a seen, but also involves an **event of seeing**

THEMATIC ROLES

(2) a. Sylvia petted Shelby.

b. Shelby spetted Sylvia. [not English, not even for a reverse meaning]

(3) If a sentence has...

- a. ...an agent, the agent will be the subject of the sentence.
- b. ...a patient and no agent, the patient will be the subject of the sentence.
- c. ...an agent and a patient, the patient will be the object of the sentence.

(4) There is an event of petting,
and this event's agent is Sylvia,
and its patient is Shelby.

(4') $\exists e[\text{Petting}(e)$ (Parsons 1990)
 $\wedge \text{Agent}(e, x)$
 $\wedge \text{Patient}(e, y)]$

(5) a. Joe hit the table with a hammer.

b. Joe hit the hammer against the table.

Proto-Roles: (Dowty 1991)

- *proto-agent entailments*: volitionality, sentience, causer, movement (relative to other participants), exists independently
- *proto-patient entailments*: undergoes change, changes portion by portion, causally affected, stationary (relative to other participants), doesn't exist independently

SETS

- (6) a. Propositions are truth-conditions, expressed as a division of possible worlds into a “true” set and a “false” set
 b. A property is a proposition missing a part.
- a property can be thought of as an association between worlds and sets — *and*
 - predication then amounts to figuring out, for each world, whether the object referred to by the subject is in the set which the predicate associates with that world

FUNCTIONS

A *function* is a mathematical object which has a set of possible “inputs” and for each input designates an “output” — specifically, we say that function consists of...

...an *argument* (the “input” — set of possible inputs: *domain*)

...and a *value* (the “output” — set of possible outputs: *range*).

Lambda calculus (λ calculus or λ abstraction)

In [computer science](#), the **lambda calculus** is a [formal system](#) designed to investigate [function](#) definition, function application, and [recursion](#). It was introduced by [Alonzo Church](#) and [Stephen Cole Kleene](#) in the [1930s](#); Church used the lambda calculus in 1936 to give a negative answer to the [Entscheidungsproblem](#). The calculus can be used to cleanly define what a [computable function](#) is. The question of whether two lambda calculus expressions are equivalent cannot be solved by a general algorithm, and this was the first question, even before the [halting problem](#), for which [undecidability](#) could be proved. Lambda calculus has greatly influenced [functional programming languages](#), especially [Lisp](#).

The lambda calculus can be called the smallest universal programming language. The lambda calculus consists of a single transformation rule (variable substitution) and a single function definition scheme. The lambda calculus is universal in the sense that any computable function can be expressed and evaluated using this formalism. It is thus equivalent to the [Turing machine](#) formalism. However, the lambda calculus emphasizes the use of transformation rules, and does not care about the actual machine implementing them. It is an approach more related to software than to hardware.

This article deals with the "untyped lambda calculus" as originally conceived by Church. Since then, some [typed lambda calculi](#) have been developed. [\[en.wikipedia.org/wiki/Lambda_calculus\]](https://en.wikipedia.org/wiki/Lambda_calculus)

- (7) a. $[\lambda x.x \text{barks}]$ c. $[\lambda f.f]$
 b. $[\lambda x.x \text{ is small}]$ d. $[\lambda x.[\lambda y.y \text{ saw } x]]$
- (8) a. $[\lambda x.x \neg \text{married} \wedge x \text{male} \wedge x \text{adult}]$ (10) a. $\lambda x \exists y [\text{love}(y, x)]$
 b. $\lambda x [\neg \text{married}(x) \wedge \text{male}(x) \wedge \text{adult}(x)]$ b. $\lambda x \exists y [\text{love}(y, x)](j)$
 c. $\exists y [\text{love}(y, j)]$
- (9) a. $\lambda x [\neg \text{married}(x) \wedge \text{male}(x) \wedge \text{adult}(x)](j)$ (11) a. $\forall x \exists y [\text{love}(x, y)]$
 b. $[\neg \text{married}(j) \wedge \text{male}(j) \wedge \text{adult}(j)]$ b. $\exists x \forall y [\text{love}(x, y)]$

REFERENCES

- Dowty, David. 1991. Thematic Proto-Roles, Argument Selection, and Lexical Semantic Defaults. *Language* 67, 547-619.
- Parsons, Terence. 1990. *Events in the Semantics of English*. Cambridge, Mass.: MIT Press.