Model Building

Johannes Dellert

Seminar für Sprachwissenschaft, Universität Tübingen

08.07.2009

What is a model?

Definition

An **interpretation** I of some formal language L consists of a non-empty set D (the **domain**) and an assignment function that maps all the function symbols and relation symbols of L onto functions and relations over D.

Definition

A **model** M is a pair $< D, I >$, where D is a set of elements (the **universe**) and I is an interpretation.

When does a formula have a model?

Definition

A model M is a **model of** a formula ϕ (written: $M \vDash \phi$) iff for every **variable assignment**, mapping the variables in ϕ to elements of M's domain, the formula evaluates to true.

Definition

If $M \vDash \phi$ holds for some model M, ϕ is called **satisfiable**.

Definition

If $M \vDash \phi$ holds for every model M, ϕ is called a **tautology**, and we write $\vDash \phi$. $\neg\phi$ is then automatically **unsatisfiable**.

A “simple” task: Model Checking

Example formula

$\forall x : (red(x) \wedge edible(x) \longrightarrow apple(x))$

Model 1

$D := \{a, b, c\}$
$GREEN(a)$
$RED(b), RED(c)$
$APPLE(A), APPLE(C)$
$EDIBLE(A), EDIBLE(C)$
$BOOK(B)$

Model 2

$D := \{a, b, c\}$
$GREEN(a), GREEN(b)$
$RED(c)$
$APPLE(b)$
$EDIBLE(a), EDIBLE(b), EDIBLE(c)$
$STRAWBERRY(a), STRAWBERRY(c)$

Infinite models

A nasty formula

$$\forall x \exists y\ r(x,y) \land \forall x\ \neg r(x,x) \land \forall x \forall y \forall z\ (r(x,y) \land r(y,z) \rightarrow r(x,z))$$

- Is this formula satisfiable?
- If yes, what is special about the model(s)?
- What would the model(s) correspond to?
- Why does this constitute a problem for a model builder?

Repetition: the purpose of theorem proving

Proof of validity (directly):

$\forall x\ (widow(x) \rightarrow woman(x)) \wedge widow(y) \rightarrow woman(y)$ ✓

Proof of unsatisfiability (via negation):

$\forall x\ (widow(x) \rightarrow woman(x)) \wedge widow(y) \wedge \neg woman(y) \Rightarrow$
$\exists x\ (widow(x) \wedge \neg woman(x)) \vee \neg widow(y) \vee woman(y)$ ✓

Proof of redundancy or entailment (via implication):

T: $\forall x\ (widow(x) \rightarrow woman(x)) \wedge widow(y)$ H: $woman(y) \Rightarrow$
$\forall x\ (widow(x) \rightarrow woman(x)) \wedge widow(y) \rightarrow woman(y)$ ✓

The purpose of model building

Proof of satisfiability (directly):

$\forall x\ (\mathit{widow}(x) \rightarrow \mathit{woman}(x)) \wedge \mathit{woman}(y) \wedge \neg\mathit{widow}(y) \Rightarrow$
produce a model full of women, none of which are widows ✓

Proof of invalidity (via negation):

$\forall x\ (\mathit{woman}(x) \rightarrow \mathit{widow}(x)) \Rightarrow \exists x\ (\mathit{woman}(x) \wedge \neg\mathit{widow}(x))$,
build such a model = a **counterexample** for original formula ✓

Proof of informativity or non-entailment (via implication):

T: $\forall x\ (\mathit{widow}(x) \rightarrow \mathit{woman}(x))$ H: $\exists y\ \mathit{widow}(y) \Rightarrow$ negation
$\forall x\ (\mathit{widow}(x) \rightarrow \mathit{woman}(x)) \wedge \forall y\ \neg\mathit{widow}(y)$ is possible ✓

Comparison: Theorem Proving vs. Model Building

Theorem Proving	Model Building
directly: show **validity**	directly: show **satisfiability**
via negation: show **unsatisfiability**	via negation: show **invalidity** (counterexample)
via implication: show **redundancy** or **entailment**	via implication: show **informativity** or **non-entailment**

- the two approaches complement each other
- theorem proving is fast but destructive and syntax-oriented
- model building is slow but constructive and semantics-oriented

Undecidability of first-order logic

Both validity and satisfiability are only **semi-decidable**:

- if there is a finite proof, the theorem prover will find it in a finite (possibly very large) number of steps
- if there is a finite model, the model builder will find it in a finite number of steps (BUT: no chance for infinite models!)
- if a formula is invalid, there is no way for the theorem prover to see this (it will explore ever more complex proofs)
- if a formula is not satisfiable, there is no way for the model builder to see this (it will construct ever bigger models)

How automated reasoning is still possible

It is still possible to build a system that can find all answers for most formulae, including most formulae relevant in linguistics!

- basic idea: start theorem prover and model builder in parallel on both the formula and its negation, see who answers first
- if formula is valid: theorem prover finds a proof
- if formula is invalid: model builder finds a counterexample
- if formula is satisfiable: model builder finds an example
- if formula is unsatisfiable: theorem prover finds a counterproof
- in a few cases: neither program terminates (timeout needed!)

Techniques: Overview

There are three major approaches to model building:

- **Hyperresolution**: very powerful and fast (but too complex to be discussed here, builds on many other techniques)
- **SEM-style**: very general, good with equations (useful for advanced group theory and other branches of mathematics)
- **MACE-style**: rather primitive, but produces minimal models (nice for NLP, delivers discourse representations "for free")

Here, we will especially discuss MACE in detail.

SEM-style Model Building

Basic procedure

- a problem consists of sorts, functions and clauses that determine constraints
- model is a definition of the functions that satisfies the axioms
- in principle, the problem is seen as a constraint satisfaction problem (CSP) and solved via backtracking
    - variables: the cells in the operation tables of the functions
    - constraints: the set of ground instances of the clauses
    - goal: a set of assignments such that the ground clauses hold
- search tree: cells are inner nodes, assignments correspond to edges, assignment propagation by constraint rewriting
- search space reduction: always choose cell with smallest number of possible values, "least number heuristic" to avoid producing isomorphic models

MACE-style Model Building

Basic procedure

- always try to find a model of some finite size m
- if no model was found, increment m and repeat the process
- resolve quantification by using conjunctions or disjunctions over the whole now finite domain
- introduce propositional variables for each predicate with all the different argument combinations
- flatten all complex structures in the clauses (such as variable inequations) to contain only shallow literals
- add more clauses to reflect functionality and totality
- apply a standard SAT-solver to the resulting propositional formula and get a set of “true” relation tuples

MACE-style Model Building

Example

Formula: $\forall x : man(x) \rightarrow (like(m,x) \wedge \neg like(b,x))$
Try to build a model with two entities: D $= \{1{,}2\}$
Propositional variables: $mi := man(i), lij := like(i,j)$
Resolve quantification and consider different assignments:
$((m1 \rightarrow (l11 \wedge \neg l11)) \wedge (m2 \rightarrow (l12 \wedge \neg l12)))\vee$
$((m1 \rightarrow (l11 \wedge \neg l21)) \wedge (m2 \rightarrow (l12 \wedge \neg l22)))\vee$
$((m1 \rightarrow (l21 \wedge \neg l11)) \wedge (m2 \rightarrow (l22 \wedge \neg l12)))\vee$
$((m1 \rightarrow (l21 \wedge \neg l21)) \wedge (m2 \rightarrow (l22 \wedge \neg l22)))$
We transform each disjunct into CNF like this:
$((m1 \rightarrow (l11 \wedge \neg l11)) \wedge (m2 \rightarrow (l12 \wedge \neg l12))) \Leftrightarrow$
$((\neg m1 \vee (l11 \wedge \neg l121)) \wedge (\neg m2 \vee (l12 \wedge \neg l12))) \Leftrightarrow$
$(((\neg m1 \vee l11) \wedge (\neg m1 \vee \neg l11)) \wedge ((\neg m2 \vee l12) \wedge (\neg m2 \vee \neg l12))) \Leftrightarrow$
$((\neg m1 \vee l11) \wedge (\neg m1 \vee \neg l11) \wedge (\neg m2 \vee l12) \wedge (\neg m2 \vee \neg l12))$

MACE-style Model Building

Example

Running a SAT solver on each of the 4 disjuncts gives us models:

1. $\neg m1 \wedge \neg m2$, i.e. $\neg man(1) \wedge \neg man(2)$
   If there are no men in the universe, the formula is satisfied.
2. $l11 \wedge \neg l21 \wedge l12 \wedge \neg l22$ $(\wedge\ e.g.\ \neg m1 \wedge m2)$
   Every entity is liked by some entity and not liked by the other.
3. $l21 \wedge \neg m1 \wedge l22 \wedge \neg l12$ $(\wedge\ e.g.\ \neg l11 \wedge m2)$
   There is one entity that is possibly a man liking himself but not liked by the other entity that is not a man (and who likes that second entity therefore does not matter).
4. $\neg m1 \wedge \neg m2$ $(\wedge\ e.g.\ \neg l11 \wedge \neg l12 \wedge \neg l21 \wedge \neg 22)$
   Isomorphic to [1], a spurious model we would want to avoid.

MACE-style Model Building

mace4: Model Building in Practice

- the `prover9-mace4` Debian package offers a nice graphical interface, with the possibility of running a theorem prover and a model builder in parallel to find a proof or a counterexample
- unfortunately, the examples only comprise mathematical problems, but also these can be very illustrative
- `mace4` can be used a console program and as a model building server that other programs can talk to
- many options, mace4 can be instructed e.g. to start at a certain model size or to proceed incrementally

Generalization to Second Order Logic

Second Order Model Building

- second order logic allows variables to range over relations (including sets) instead of just individuals
- second order model theory is rather complex and involved (problems already start with the question whether all possible relations over the individual elements in the domain are also included in the domain)
- however, with enormous costs, it is possible to describe second-order structures in first-order logic (use e.g. `isXaryRel`, `elementOf`, `isIndividual` as first-order predicates and introduce new axioms)
- for very small models, this is a viable option and allows model building for formulae of second-order logic (KIMBA)

Applications: Overview

Model Building in NLP

- first-order logic is not expressive enough to elegantly cover many semantic phenomena (e.g. quantifiers such as “most”, reciprocity, plural semantics)
- however, already restricted forms of second-order logic suffice to express most of these phenomena
- for such restricted variants, model building is still computationally viable (or will be in a few decades)
- a very new field, most work that is motivated by linguistics is exploratory work by Karsten Konrad
- his results have not yet been taken up by the RTE community

Gardent & Konrad 2000: Interpreting Definites

Claim

Model builders can be used to generate procedural interpretations for natural language semantics, and they are specifically useful in anaphoric linking, bridging and accomodation.

Example

Singular definite DPs such as “the rabbit” in
“A rabbit sleeps. The rabbit has a nightmare” or
“Bugs and Bunny are rabbits. Bugs is in the hat. John removes the rabbit from the hat.”

Gardent & Konrad 2000: Basic idea

Approach

Take a sentence or discourse, convert it into a logical formula, give the formula to a model builder and have it build the minimal model of the discourse. This model often comes close to intuitive linguistic interpretation.

Example

Sentence: Bugs likes a white female rabbit.
Formula: $\exists x : rab(x) \land white(x) \land fem(x) \land like(bugs, x)$
Model 1: $\{rab(bugs), like(bugs, bugs), white(bugs), fem(bugs)\}$
+ World knowledge: $rab(bugs) \land \neg fem(bugs)$
Model 2: $\{rab(bugs), rab(c_1), like(bugs, c_1), white(c_1), fem(c_1)\}$

Gardent & Konrad 2000: Semantics of Definites

Definition

$[\mathit{the}] = \lambda P_1 \lambda P_2 \lambda P_3 . |P_1 \cap P_2| = 1 \wedge P_1 \cap P_2 \subseteq P_3$,
where P_1 is the common-noun property, P_2 the identifying property, and P_3 the property denoted by the rest of the sentence.

Remarks

- note that the definition of the semantics for "the" contains second-order variables and set-theoretic operations
- for this reason, we need a higher-order model builder to directly construct models from such formulae
- Gardent & Konrad use the KIMBA system for this purpose.

Gardent & Konrad 2000: Semantics of Definites

Example

Sentence: The red rabbit sleeps.
Common-noun property: $red \cap rab$
Identifying property: $\lambda x \ true$, provided there is a unique red rabbit in the context (which we assume here)
VP-property: $sleep$
Semantic representation: $THE(red \cap rab)(\lambda x \ true)(sleep)$
Applying the definition:
$|(red \cap rab) \cap (\lambda x \ true)| = 1 \wedge (red \cap rab) \cap (\lambda x \ true) \subseteq sleep$
Simplify: $|red \cap rab| = 1 \wedge red \cap rab \subseteq sleep$

Gardent & Konrad 2000: Applications

Example (Anaphoric Linking)

Sentence: A rabbit sleeps. The rabbit dreams.
Semantic representation:
$\exists x(rab(x) \wedge sleep(x)) \wedge THE(rab)(\lambda x\ true)(dream)$
Model generated: $\{rab(c_1), sleep(c_1), dream(c_1)\}$

Example (Bridging)

Sentence: A rabbit sleeps. The tail is twitching.
LF: $\exists x(rab(x) \wedge dream(x)) \wedge THE(tail)(\lambda x\ true)(twitch)$
World knowledge:
$\forall x(rab(x) \rightarrow \exists y : tail(y) \wedge of(x, y)) \wedge \forall x(tail(x) \rightarrow \neg rab(x))$
Model generated:
$\{rab(c_1), sleep(c_1), tail(c_2), twitch(c_2), of(c_1, c_2)\}$

Gardent & Konrad 2000: Applications

Example (Accomodation)

Sentence: The rabbit eats the carrot.
LF: $THE(rab)(\lambda x\ true)(\lambda x\ THE(crt)(\lambda x\ true)(\lambda z\ eat(x,z)))$
Model generated: $\{rab(c_1), crt(c_2), eat(c_1,c_2)\}$

Example (Uniqueness)

Discourse: Bugs and Bunny are rabbits. Bugs is in the hat. John removes the rabbit from the hat.
Semantic representation for Sentence 3:
$THE(rab)(\lambda x\ inHat(x))(\lambda y\ THE(hat)(\lambda x\ true)(\lambda x\ rmv(j,y,x)))$
Model generated:
$\{rab(bugs), rab(bunny), inHat(bugs), rmv(j, bugs, hat)\}$

Gardent & Konrad 2000: Conclusion

Conclusions

- model building offers a unified approach covering both coreference and accomodation in a natural way
- main reason: models generated by model builders represent **locally minimal models of discourse**
  $\longrightarrow$ assume coreference if possible, accomodate if necessary
- the style of reasoning that results in coverage of some bridging phenomena is similar to **abduction**
- approach must be extended to deal with conditional and negative statements
- ongoing research focuses on further narrowing down possible models according to linguistic constraints, possibly resulting in more efficient model building algorithms for NLP

Conclusion

Model building

- tries to find a formal structure in which a formula holds
- complements theorem proving in computational semantics
- approaches usually rely on constraint satisfaction for propositional atoms over finite domains

(Potential) Application to RTE

- used in generating models of discourse domains
- models tend to be minimal, which is useful for modeling coreference, bridging and accomodation in natural language
- second-order model building necessary for NLP

References I

[1] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI Publications, Stanford, California, 2005.

[2] Claire Gardent and Karsten Konrad. *Interpreting Definites Using Model Generation*. 2000.

[3] Karsten Konrad. *System Description: Kimba, A Model Generator for Many-Valued First-Order Logics*, 1999.

[4] Karsten Konrad. *Model Generation for Natural Language Interpretation and Analysis*. PhD thesis, 2000.

[5] Jian Zhang and Hantao Zhang. *Generating Models by SEM*. 1996.

The End

Thank you!