MODEL CHECKING AND FINITE-MODEL THEORY IN THE DIFFERENT CONTEXTS OF COMPUTATIONAL LOGICS, KNOWLEDGE REPRESENTATION, AND DATABASE THEORY

MOSHE Y. VARDI

RICE UNIVERSITY

THE TRUTH,

THE WHOLE TRUTH,

AND NOTHING BUT THE TRUTH

MOSHE Y. VARDI

RICE UNIVERSITY

MATH AND SCIENCE

Galileo: "The book of nature is written in the language of mathematics"

Wigner, 1963: "On the unreasonable effectiveness of mathematics in the physical sciences"

question: why is mathematics so effective?

BACK TO BASICS

- All men are mortal
- Socrates is a man
- Therefore, Socrates is mortal

Forms: $a \in P$: membership

$P \subseteq Q$: containment

Reasoning:

- Model checking: $M \in models(\varphi)$
- Validity checking: $models(k) \subseteq models(\varphi)$

The Basic Reduction: TFAE

- $models(k) \subseteq models(\varphi)$
- $models(k) - models(\varphi) = \phi$
- $models(k) \cap models(\neg\varphi) = \phi$
- Not: $\exists M, M \in models(K), M \in models(\neg\varphi)$

LE

Let there Be M!

Three fundamental techniques:

1. Canonical model Property
2. Bounded model Property
3. Tree model Property

LOGIC AND COMPUTER SCIENCE

Halpern, Kolaitis, Papadimitriou, Vardi, Vianu, 1999: "On the Unreasonable effectiveness of logic in computer science" (AAAS Symp.)

IFCOLOG: International Federation of Computational Logicians (an umbrella organization, covering 10,000 members)

Question: Why is logic so effective?

Reminder: For its first 2,400 years logic was part of philosophy.

LOGIC AND AI

McCarty, 1958: Reasoning = Logic

- Knowledge representation:
  collection K of sentences
- Knowledge generation:
  Deduction: $K \vdash \varphi$

Difficulties:

- Knowledge representation requires powerful logics
- powerful logics make deduction hard
- classical deduction is monotonic

Claim: Very limited success, inspite of 40 years of research

Impact of Logic

Question: what concrete, significant impact did logic have on computing technology?

Answers:

- Propositional logic: circuits
- First-order logic: database queries

SQL

SQL: essentially 1st-order

- conceived as user-friendly
- used as an inter-DBMS language

Example: list pairs of persons such that each directed the other in a movie.

Movies(X, Y): X directed Y

- FOL: $Movies(X,Y) \wedge Movies(Y,X)$
- SQL:

```
Select m1.director
       m2.actor
From   Movies m1
       Movies m2
Where  m1.director = m2.actor
and    m1.actor = m2.director
```

Formulas as Queries

$\varphi(x_1, \ldots x_n)$: formula

$\mathcal{B}$: relational structure

$$\varphi(\mathcal{B}) = \{\langle a_1, \ldots a_n \rangle : \mathcal{B} \models \varphi(a_1, \ldots a_n)\}$$

Key points:

- Based on Tarski's notion of truth
- Codd elevated the status of formulas

MODEL CHECKING

The basic idea:

- Represent a finite-state program as a finite structure
- check that the structure is a model of the spec [CES, QS]

Success:

- [AK85]: "one of the most exciting development in the theory of program correctness"
- 1981-1995: the "push" period
- 1995- : the "pull" period

Formulas as Specs

φ : temporal formula

K : kripke structure

$\varphi(K) = \{ w : K, w \models \varphi \}$

key points:

- Based on kripke's notion of truth
- Pnueli proposed using temporal logic to specify ongoing behavior

The Search for the Truth

Truth: satisfaction in one structure

validity: satisfaction in all structures

claims:

- In database query evaluation and in temporal model checking the focus is on truth.
- Automated reasoning should include truth checking, in addition to validity checking.

VALIDITY IS IMPORTANT

Examples:

- Query optimization:

  Is $Q_1 \equiv Q_2$?

- Spec correctness:

  Is φ valid?

Observation: very different algorithms for truth and validity checking.

Question: Can we unify truth and validity checking?

TRANSITION SYSTEMS

$T = (S, S_0, \Lambda, O, \Pi)$ kripke structure

S: states

$S_0 \subseteq S$: initial states

$\Lambda \subseteq S^2$: transition relation

O: observables

$\Pi : S \to 2^O$: observation function

key insight: every discrete system can be modeled as a labeled graph.

Temporal Specs

Temporal specs: description of ongoing behavior

Examples:

- process 1 and process 2 can not enter the critical section at the same time (safety)
- Every request will be granted (liveness)
- Every continous request will be granted (liveness)
- Every repeated request will be granted (liveness)

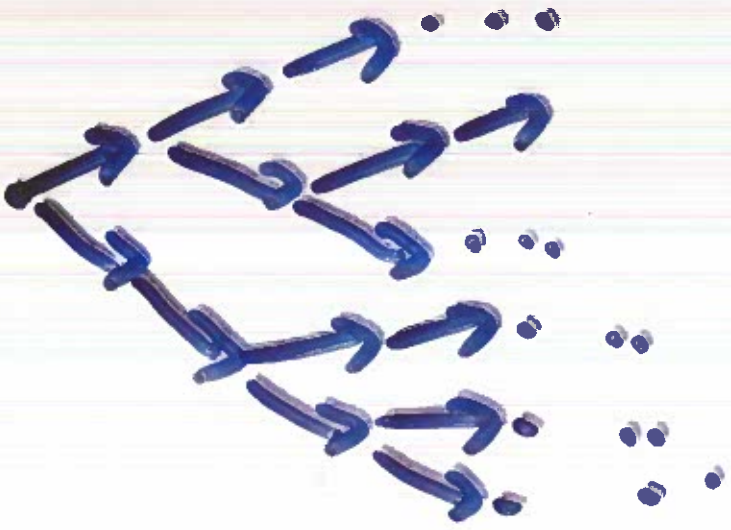
Linear vs. Branching

Linear time: a program generates a set of computations

Specs: describe computations

Branching time: a program generates a computation tree

Specs: describe computation trees



!!

PROGRAM EQUIVALENCE

P_1:

P_2:

Linear: $P_1 \equiv P_2$

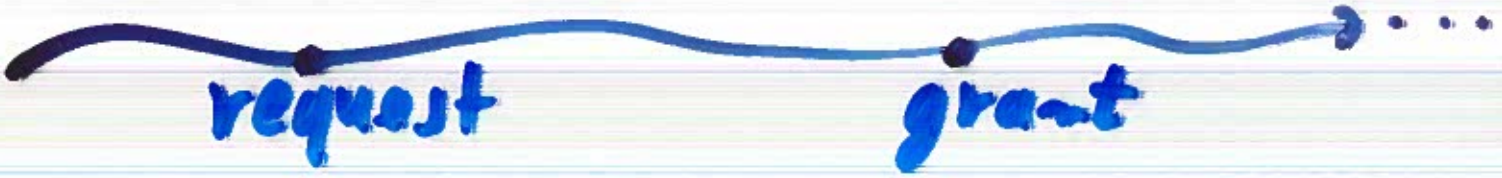
Branching: $P_1 \not\equiv P_2$

Temporal Logics

Linear Temporal Logic (LTL):

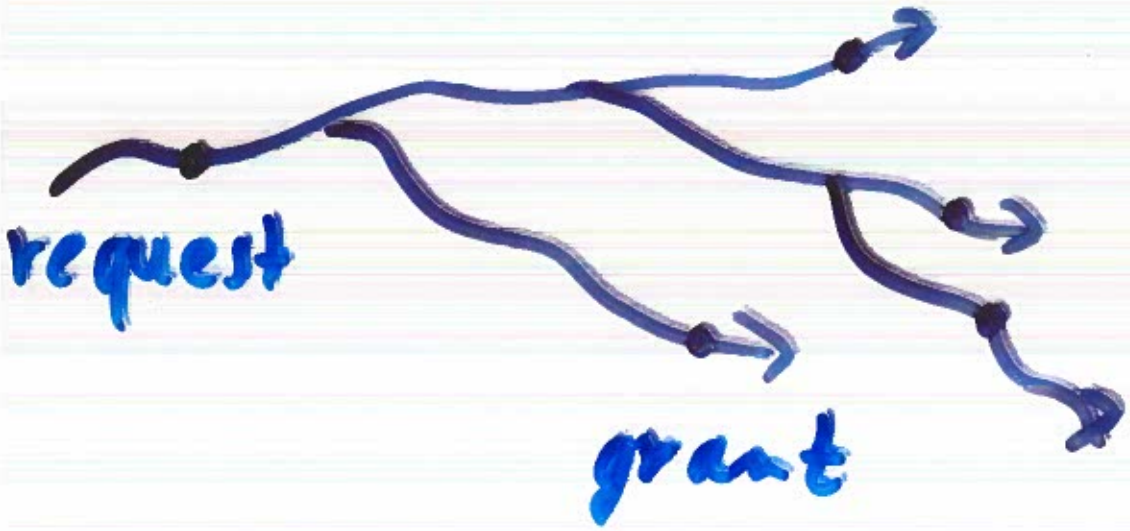
always (request → eventually grant)

Every request is eventually granted



Computation-Tree Logic (CTL):

∀ always (request → ∀ eventually grant)



TRUTH vs. VALIDITY

TRUTH: $\mathcal{M} \models \varphi$

↑ one structure

VALIDITY: $\models \varphi$

↑ all structures

Traditional focus:
Necessary truth = validity

Completeness Theorem:
$K \models \varphi$ iff $K \vdash \varphi$

TRUTH + VALIDITY

φ: spec $|\varphi| = m$

T: system $|T| = m$

Upper Time Bounds:

	LTL	CTL	
truth	$m \cdot 2^m$	$m \cdot m$	LP CES
validity	2^m	2^m	SC EH

Key Points

1. Low complexity in m
2. Different complexity in m for CTL.

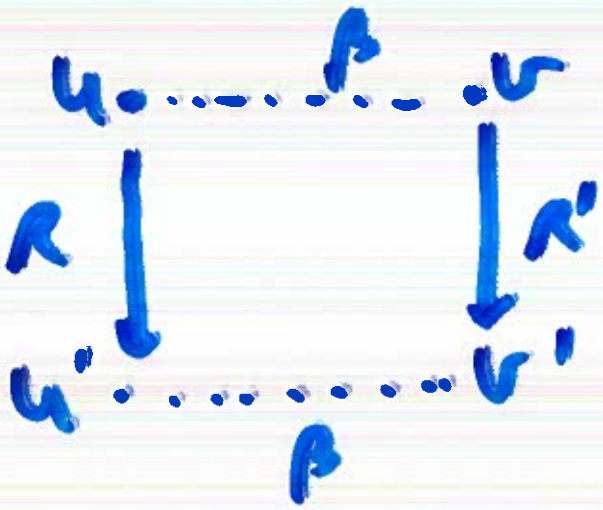
BISIMULATION

$T = (S, R, O, \pi)$

$T' = (S', R', O, \pi')$

Bisimilarity: $\beta \subseteq T \times T'$

$\beta(u, v) \Rightarrow$

- $\pi(u) = \pi'(v)$
- 

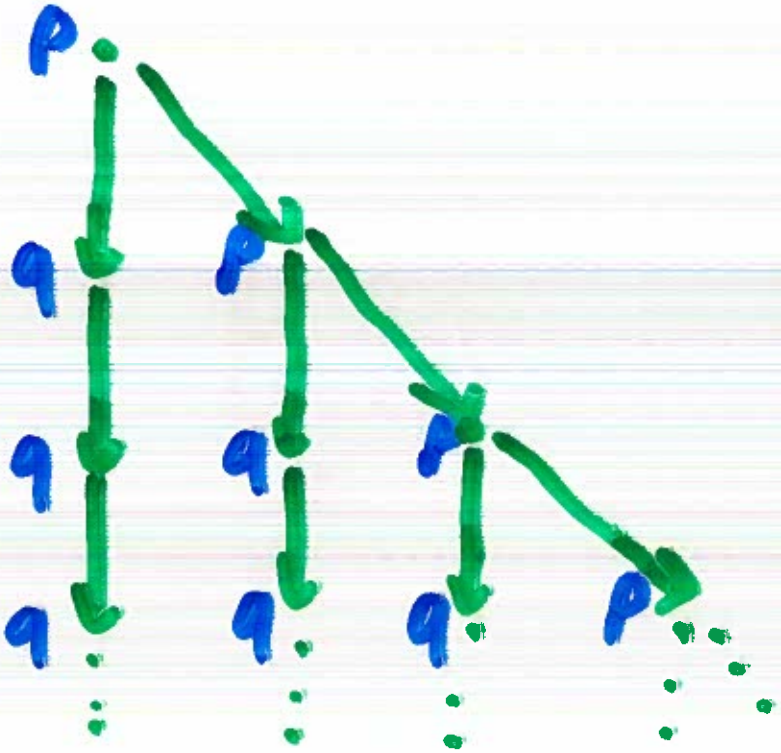
Fact: $\beta(u, v) \Rightarrow$

$T, u \models \varphi$ iff $T, v \models \varphi$

for $\varphi \in ML, LTL, CTL$

TREE MODEL PROPERTY

Fact: A transition system is bisimilar to its unfolding



Corollary:

- $T \Vdash \varphi$ iff $\mathrm{unfold}(T) \Vdash \varphi$.
- $\Vdash \varphi$ iff φ is true in all $|\varphi|$-ary trees.

Linear Temporal Logic

LTL: logic of temporal sequences

Main feature: time is implicit

- next φ: φ holds next
- eventually φ: φ holds eventually
- always φ: φ holds always
- φ until ψ: φ holds until ψ holds

Examples

- always($\neg cs_1 \vee \neg cs_2$):
  safety
- always(request → eventually grant)
  liveness
- always(request → request until grant)
  liveness
- always eventually request → eventually grant
  liveness

Semantics

Computation: $\sigma : \mathbb{N} \to 2^{\mathcal{O}}$

- $\sigma \models p$ if $p \in \sigma(0)$
- $\sigma \models \text{next } \psi$ if $\text{tail}(\sigma) \models \psi$
- $\sigma \models \text{always } \psi$ if $\text{tail}^i(\sigma) \models \psi$ for $i \geq 0$

$T = (S, S_0, R, \mathcal{O}, \Pi)$

Run: $w_0, w_1, w_2, \ldots$

- $w_0 \in S_0$
- $(w_i, w_{i+1}) \in R$

Computation: $\Pi(w_0), \Pi(w_1), \ldots$

- $L(T)$: computations of T
- $T \models \psi$: all computations of T satisfy ψ

BÜCHI AUTOMATA

$A = (\Sigma, S, S_0, \rho, F)$

- Σ : alphabet
- S : states
- $S_0 \subseteq S$: initial states
- $\rho \subseteq S \times \Sigma \times S$: transition relation
- F : accepting states

Input: $a_0, a_1, \ldots$

Run : $s_0, s_1, s_2 \ldots$

- $s_0 \in S_0$
- $(s_i, a_i, s_{i+1}) \in \rho$

Acceptance: F visited i.o.

$L(A)$: lang. accepted by A

Properties

- **Closure:** Let A_1, A_2 be Büchi automata w. m_1, m_2 states. There is a Büchi automata $A_1 \times A_2$ w. $2m_1m_2$ states s.t. $L(A_1 \times A_2) = L(A_1) \cap L(A_2)$

- **Emptiness:** There is a linear-time algorithm that checks whether $L(A) = \emptyset$.

LTL vs. AUTOMATA

Compilation Thm: Given an LTL formula φ, we can construct a Büchi automaton A_φ w. $2^{O(|\varphi|)}$ states such that σ satisfies φ iff A_φ accepts σ.

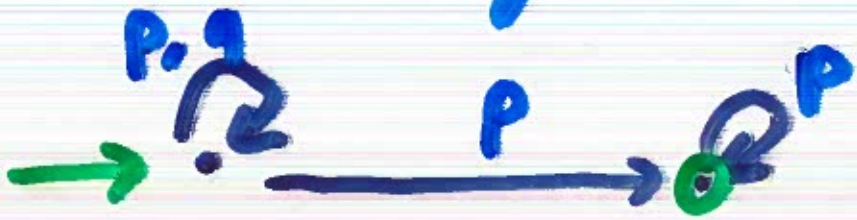
Application:

validity checking: φ is valid iff $L(A_{\neg\varphi}) = \emptyset$ - yields an algorithm of time $2^{O(|\varphi|)}$

Examples

- always eventually p:

- eventually always p:



TRUTH CHECKING

T.F.A.E:

- $T \models \varphi$
- $L(T) \subseteq L(A_\varphi)$
- $L(T) - L(A_\varphi) = \emptyset$
- $L(T) \cap L(A_{\neg\varphi}) = \emptyset$
- $L(T \times A_{\neg\varphi}) = \emptyset$

Application: Truth checking algorithm of time $|T| \cdot 2^{O(|\varphi|)}$

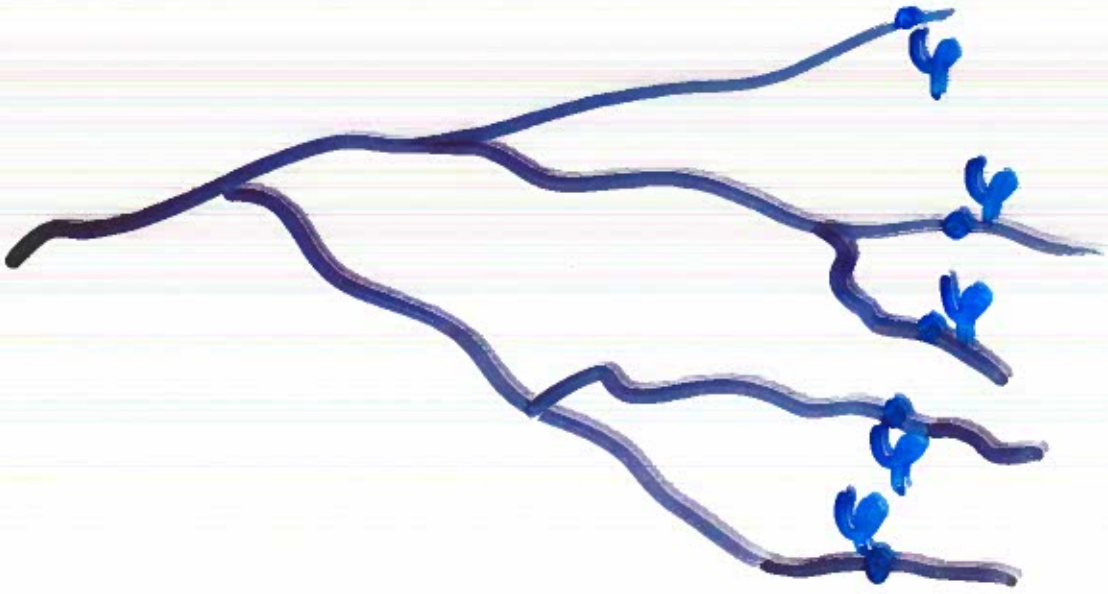
Bottom line: Büchi automata unify truth and validity checking for LTL.

COMPUTATION TREE LOGIC

CTL: logic of temporal trees

Main feature: quantification over possible futures

- $\forall$ always φ: φ holds always
- $\forall$ eventually φ: φ is inevitable
- $\exists$ always φ: always φ is possible
- $\exists$ eventually φ: φ is possible



BÜCHI TREE AUTOMATA

$A = (\Sigma, S, S_0, \rho, F)$

- Σ, S, S_0, F : same
- $\rho \subseteq S \times \Sigma \times S \times S$: transition relation

Input: $\tau : \{0,1\}^* \to \Sigma$

Run : $r : \{0,1\}^* \to S$

- $r(\varepsilon) \in S_0$
- $(r(x), \tau(x), r(x0), r(x1)) \in \rho$

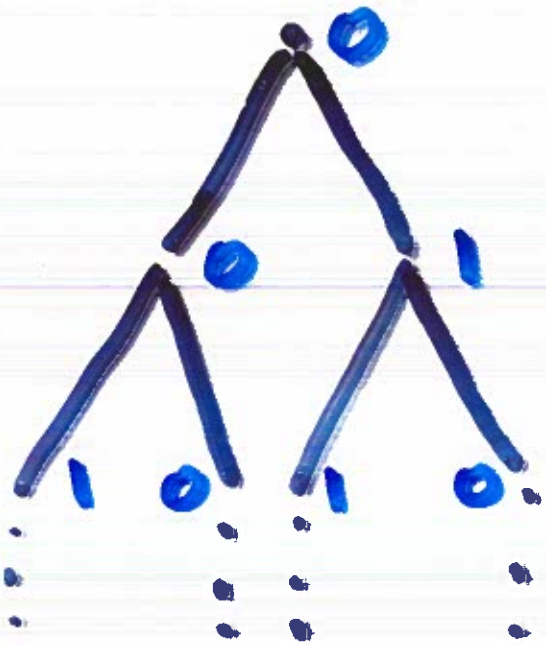
Acceptance: F visited i.o. along every branch

L(A): Tree language of A

n

$t \quad u$

- $n \in S_0$
- $(n, o, t, u) \in P$



PROPERTIES

- closure under intersection
- Quadratic-time emptiness checking algorithm (i.e., $L(A) = \emptyset$?)

CTL vs. AUTOMATA

Compilation Thm: Given a CTL formula φ, we can construct a Büchi tree automaton A_φ w. $2^{O(|\varphi|)}$ states such that τ satisfies φ iff A_φ accepts τ.

Application:

Validity checking: φ is valid iff $L(A_{\neg\varphi}) = \emptyset$ - yields an algorithm of time $2^{O(|\varphi|)}$

CTL TRUTH CHECKING

Problem: CTL truth checking is linear in $|\varphi|$, but $A\varphi$ is exponential in $|\varphi|$.

Solution: alternation

non determinism: $\exists$-choice

- $\exists$-choice: some run accepts
- $\forall$-choice: all runs accept

Alternation: $\exists + \forall$-choice

NONDETERMINISM VS. ALTERNATION

Nondeterminism:

- $(q, a) \longmapsto \{q_1, q_2, q_3\}$
- $\delta(q, a) = q_1 \vee q_2 \vee q_3$

Alternation:

$$\delta(q, a) = (q_1 \wedge q_2) \vee (q_3 \wedge q_4)$$

In general:

$$\delta : S \times \Sigma \to B^+(S)$$

positive formulas on S

LTL vs. AUTOMATA

Compilation Thm: Given an LTL formula φ, we can construct an alternating Büchi automaton A^a_φ w. $O(|\varphi|)$ states such that σ satisfies φ iff A^a_φ accepts σ.

Fact: There is an exponential translation from alt. Büchi automata to nondet. Büchi automata.

Application: LTL $\xrightarrow{\text{exp}}$ nondet. Büchi automata

Example

φ: always eventually p

φ': eventually p

$$A^a_\varphi = (\Sigma, S, S_0, \rho, F)$$

$$\Sigma = 2^{\{p\}}, \quad S = \{\varphi, \varphi'\}, \quad S_0 = F = \{\varphi\}$$

$$\rho(\varphi, p) = \varphi$$

$$\rho(\varphi, \bar{p}) = \varphi \wedge \varphi'$$

$$\rho(\varphi', p) = \text{true}$$

$$\rho(\varphi', \bar{p}) = \varphi'$$

ALT. TREE AUTOMATA

Non determinism:

- $(q, a) \mapsto \{\langle q_1, q_2\rangle, \langle q_3, q_4\rangle\}$
- $\rho(q, a) = [(1, q_1) \wedge (2, q_2)] \vee [(1, q_3) \wedge (2, q_4)]$

Alternation:

$\rho(q, a) = (1, q_1) \vee [(2, q_2) \wedge (2, q_3)]$

In general:

$\rho : S \times \Sigma \to B^+(\{1, 2\} \times S)$

CTL vs. AUTOMATA

Compilation Theorem: Given a CTL formula φ, we can construct an alt. Büchi tree automaton A^a_φ w. $O(|\varphi|)$ states such that τ satisfies φ iff A^a_φ accepts τ.

Fact: Emptiness problem for alt. Büchi tree auto. is EXPTIME-complete.

- good for validity checking
- bad for truth checking

TRUTH CHECKING

T. F. A. E:

- $T \models \varphi$
- $\text{Unfold}(T) \in L(A^q_\varphi)$
- $L(T \times A^q_\varphi) \neq \emptyset$

Crux: $T \times A^q_\varphi$ is on a 1-letter alphabet

Fact: There is a quadratic time alg. that checks emptiness of 1-letter aut.
Büchi: tree automata.

Application: Quadratic truth checking algorithm.

BETTER COMPLEXITY

Fact: If φ is a CTL formula, then A^a_φ is a weak alternating automaton [MS]

Fact: There is a linear-time algorithm that checks emptiness of 1-letter WAA.

Application: linear-time truth checking algorithm for CTL.

ultimate Unification

- Truth checking: 1-letter emptiness
- validity checking: 2-letter emptiness

Unification:

2-letter emptiness is solved via a (nontrivial) reduction to 1-letter emptiness

In Conclusion

1. Tree-model property is often more useful than finite-model property.

2. Tree automata are useful in automated reasoning.

3. Truth and validity checking can be unified.

Future work: other examples of such unification.

LTL vs. CTL

- eventually always p:

  "p is ultimately high"

- ∀ eventually ∀ always p:

  "p will stabilize high after a bounded number of steps"

Benefit: Nonmonotonicity

Domain — Relations

$\mathcal{M}_1 = (D_1, R^1_1, \ldots R^1_k)$

$\mathcal{M}_2 = (D_2, R^2_1, \ldots, R^2_k)$

$\mathcal{M}_2$ substructure of $\mathcal{M}_1$: $\mathcal{M}_2 \subseteq \mathcal{M}_1$

- $D_2 \subseteq D_1$
- $R^2_i = R^1_i | D_2$

Well-known:

- $\mathcal{M}_1 \models \varphi \not\Rightarrow \mathcal{M}_2 \models \varphi$
- $\mathcal{M}_2 \models \varphi \not\Rightarrow \mathcal{M}_1 \models \varphi$

Truth is not monotonic!

Benefit: Logical Omniscience

Hintikka, 1958: possible-worlds approach to epistemic reasoning -

A knows φ if φ holds in all worlds that A considers to be possible

Consequence: logical omniscience -

A knows all consequences of her knowledge, including all tautologies

Is this problematic?

Suppose A has 3 possible worlds, then knowing all tautologies is easy!

MODEL-CHECKING MANIFESTO

Halpern + Vardi, 1991:

Reasoning = Truth Checking =
Model Checking

Knowledge Representation:

semantic structure M

Knowledge generation:

Model checking $M \models \varphi$

Example: muddy children

Claim: Propositial satisfiability and model checking have so far been the only successful modes of automated reasoning

CONJUNCTIVE QUERIES

Select-Project-Join queries:

$FO(+, \wedge, \exists)$:

- Positive atomic formulas
- Conjunction
- Existential quantification

Example:

grandparent(x,y) :- PARENT(x,z), PARENT(z,y)

Key notion: $Q_1 \sqsubseteq Q_2$: for each db B,

$Q_1(B) \subseteq Q_2(B)$

Equivalently: $Q_1 \models Q_2$

Note: $Q_1 \equiv Q_2$ iff $Q_1 \sqsubseteq Q_2$ & $Q_2 \sqsubseteq Q_1$

CANONICAL MODEL PROPERTY

Chandra + Merlin, 1976:

Conjunctive Query ≈ Relational DB

$$CQ \longmapsto B^{Q}$$

Example: Grandparent(x, y) :- Parent(x, z), Parent(z, y)

B^{Q} :

Parent	
x	z
z	y

Thm: [CM'76]

$$Q_1 \models Q_2 \iff B^{Q_1} \models Q_2$$

$Q_1 \models Q_2$: logical implication

$B^{Q_1} \models Q_2$: model checking

canonical model: B^{Q_1}

BOUNDED MODEL PROPERTY

FOL^k: FOL with k variables

Fact: many description logics $\subseteq FO^2$

Validity checking:

FOL: undecidable

FOL^3: undecidable

FOL^2: co-NEXPTIME-complete

Theorem: [Grädel, Kolaitis, Vardi, 1997]
If an FO^2 formula is satisfiable, then it has a model of exponential size.

TREE-Model property

PA: Presburger Arithmetics = Theory of natural addition

$$\forall x \forall y \forall z ((x+y)+z = x+(z+y))$$

$$models(\varphi(x,y,z)) = \{(a,b,c) : \mathbb{N} \models \varphi(a,b,c)\}$$

$$Language(\varphi(x,y,z)) = \{ a_0 b_0 c_0 \, a_1 b_1 c_1 \ldots a_n b_n c_n : \mathbb{N} \models \varphi(a,b,c) \}$$

$a_0 a_1 \ldots a_n$: binary encoding of a
$b_0 b_1 \ldots b_n$: binary encoding of b
$c_0 c_1 \ldots c_n$: binary encoding of c

Theorem: [Cobham, 1965]

Language $(\varphi(x,y,z))$ is regular

Proof: Induction: constructive proof

$L(A_\varphi) = Language(\varphi)$

A_φ : automaton

Decidability

Algorithm for checking satisfiability:

1. Construct A_φ
2. check $L(A_\varphi) \neq \emptyset$

Nonemptiness Test:

$A = (\Sigma, S, S_0, \rho, F)$

Σ: alphabet

S: states

$S_0 \subseteq S$: initial states

$\rho : S \times \Sigma \to 2^S$: transition function

$F \subseteq S$: accepting states

$G_A = (S, E_A)$, $E_A = \{(s,t) : t \in \rho(s,a)\}_{a \in \Sigma}$

Fact: $L(A) \neq \emptyset$ iff there is a path from S_0 to F in G_A.

MEMBERSHIP TEST

Membership problem: $A = (\Sigma, S, S_0, \rho, F)$

$w \in \Sigma^*$

$w \stackrel{?}{\in} L(A)$

Let $w = a_0 a_1 \dots a_{n-1}$

Define: $A^w = (\{a\}, S \times \{0, \dots, n\}, S_0 \times \{0\}, \rho^w, F \times \{n\})$

$$\rho^w((s,i),a) = \{(t, i+1) : t \in \rho(s, a_i)\}$$

Intuitively: A^w is A "specialized" to w.

Lemma: $w \in L(A)$ iff $L(A^w) \neq \emptyset$

Moral: Emptiness testing and membership testing coincide for FSA.

Next: Temporal reasoning from this perspective.