Automated Reasoning

Hank Hwang

自我介紹

- 黃煜翔(Hank Hwang)
- 台大資工、數學雙主修畢業
- 杏雅各生醫科技創辦人
- Mathematical Logic
- Finite Model Theory and Descriptive Complexity theory
- Computability Theory and Algorithmic Randomness
- Modal Logic and Epistemic Modal Logic
- Model Checking and Formal Verification
- Automatic Theorem Proving
- Automata Theory
- Formal Epistemology

What is reasoning?

- Deductive Reasoning(演繹推理)
- Inductive Reasoning(歸納推理)
- Abductive Reasoning(溯因推理)

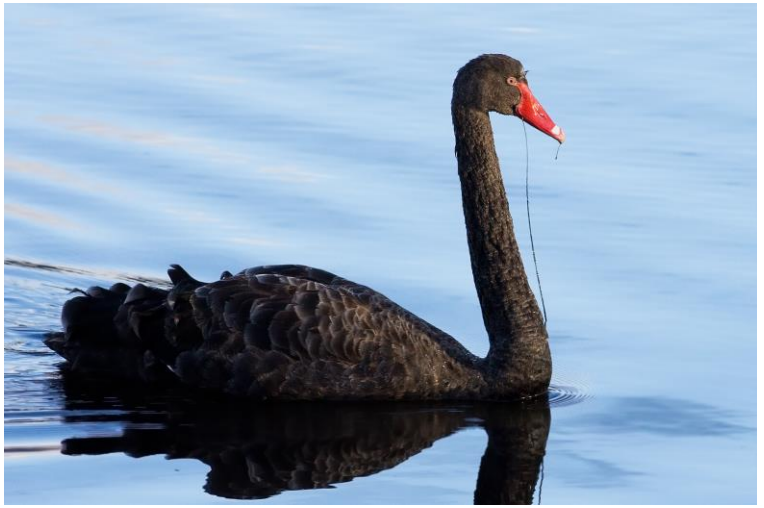
Deductive Reasoning(演繹推理)

- 台灣所有的狸貓都是棕色的
- x是狸貓且x來自台灣
- x是棕色的

黑天鵝

- “在18世紀歐洲人發現澳洲之前，由於他們所見過的天鵝都是白色的，所以在當時歐洲人眼中，天鵝只有白色的品種。"

- “直到歐洲人發現了澳洲，看到當地的黑天鵝後，人們認識天鵝的視野才打開，只需一個黑天鵝的觀察結果就能使從無數次對白天鵝的觀察中推理出的一般結論失效”
- ----Wikipedia



Inductive Reasoning(歸納推理)

- 前提:
- x是狸貓，x來自台灣，x是棕色的。
- y是狸貓,y來自台灣,y是棕色的。
- z是狸貓,z來自台灣,z是棕色的。
- …….
- ……
- 結論:
- 所有來自台灣的狸貓都是棕色的。

Raven Paradox

- 所有烏鴉都是黑色的。(P1)
- 所有不是黑的東西不是烏鴉（P2)
- Note that P1 and P2 are logically equivalent!
- 看到一隻黑烏鴉，增強(P1)的可能性。
- 看到一顆紅頻果，增加(P2)的可能性！？？！！！

Abductive Reasoning(溯因推理)

- 台灣的狸貓都是棕色的
- 美美這隻狸貓是棕色的
- 美美來自台灣

First-Order Logic (FOL)

- **Constants:** **Alice,Bob,4 ...**
- **Variables:** **x, y, a, b,...**
- **Predicates:** **Person(John), Siblings(Alice, Bob), IsOdd(4), ...**
- **Functions:** **MotherOf(John), Sqrt(x), ...**
- **Connectives:** $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- **Equality:** **=**
- **Quantifiers:** $\forall, \exists$

- **Term:** **Constant** or **Variable** or **Function($Term_1$, ... , $Term_n$)**
- **Atomic sentence:** **Predicate($Term_1$, ... , $Term_n$)** or **$Term_1$ = $Term_2$**
- **Complex sentence:** made from atomic sentences using connectives and quantifiers

FOL Syntax: Quantifiers

Universal quantifier: ∀<variable> <sentence>

- Means the sentence is true **for all** values of *x* in the domain of variable *x*

- Main connective typically ⇒ forming if-then rules
  - **All humans are mammals** **becomes in FOL:**

    ∀x Human(x) ⇒ Mammal(x)

    i.e., for all *x*, if *x* is a human then *x* is a mammal
  - **Mammals must have fur** **becomes in FOL:**

    ∀x Mammal(x)⇒ HasFur(x)

    for all *x*, if *x* is a mammal then *x* has fur

FOL Syntax: Quantifiers

$\forall x$ (Human(x) $\Rightarrow$ Mammal(x))

- Equivalent to the conjunction of instantiations of x:

  (Human(Jim) $\Rightarrow$ Mammal(Jim)) $\wedge$
  (Human(Deb) $\Rightarrow$ Mammal(Deb)) $\wedge$
  (Human(22) $\Rightarrow$ Mammal(22)) $\wedge$ …

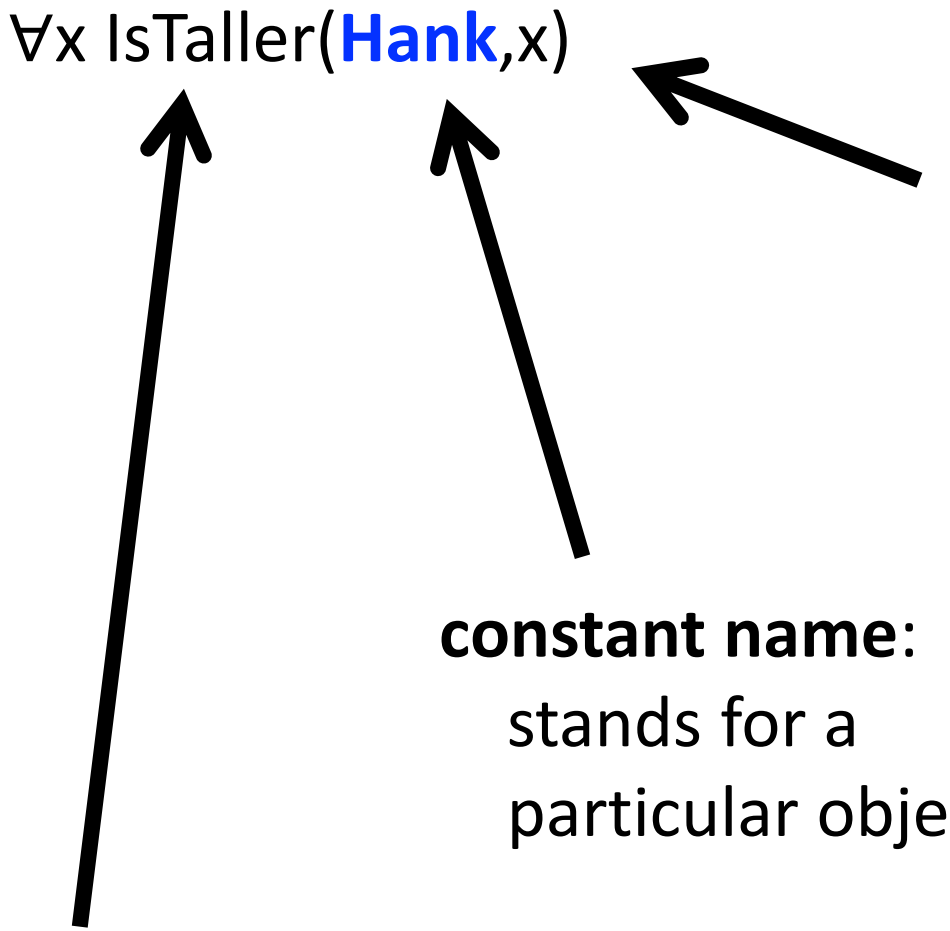
FOL Syntax: Quantifiers

Existential quantifier: ∃<variable> <sentence>

- Means the sentence is true
  for some value of *x* in the domain of variable *x*

- Main connective is typically ∧
  - **Some humans are old** **becomes in FOL:**
  - **∃x Human(x) ∧ Old(x)**
    there exist an *x* such that *x* is a human and *x* is old
  - **Mammals may have arms.** **becomes in FOL:**
  - **∃x Mammal(x) ∧ HasArms(x)**
    there exist an x such that x is a mammal and x has arms

"Hank is taller than everyone":



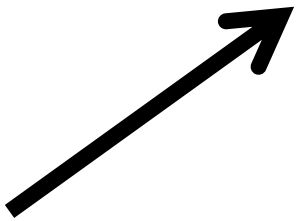
variable:
stands for an object (person)

constant name:
stands for a particular object

relation name:
stands for a mapping, object(s) ↦ **T/F**

“Hank’s dad is taller than everyone else’s dad”:

$\forall x\ (\neg(x=\textbf{Hank}) \rightarrow \text{IsTaller}(\text{Father}(\textbf{Hank}),\text{Father}(x)))$



function name:

stands for a mapping,

object(s) $\mapsto$ object

De Morgan's Law for Quantifiers

De Morgan's Rule

$$P \wedge Q \equiv \neg(\neg P \vee \neg Q)$$

$$P \vee Q \equiv \neg(\neg P \wedge \neg Q)$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

Generalized De Morgan's Rule

$$\forall x\, P \equiv \neg\exists x\, (\neg P)$$

$$\exists x\, P \equiv \neg\forall x\, (\neg P)$$

$$\neg\forall x\, P \equiv \exists x\, (\neg P)$$

$$\neg\exists x\, P \equiv \forall x\, (\neg P)$$

Example

- Friends of friends are friends
- $\forall x \forall y \forall z \; Fr(x, y) \wedge Fr(y, z) \Rightarrow Fr(x, z)$
- Friendless people smoke.
- $\forall x \; (\neg(\exists y \; Fr(x, y)) \Rightarrow Sm(x))$
- “If two people are friends, then either both smoke or neither does.”
- $\forall x \forall y \; Fr(x, y) \Rightarrow (Sm(x) \Leftrightarrow Sm(y))$

FOL and Database

FOL	DB
friends	`CREATE TABLE FRIENDS (friend1 : INTEGER` `friend2 : INTEGER)`
$friends(x, y)$	`SELECT friend1 AS x friend2 AS y` `FROM FRIENDS`

$friends(x, x)$	`SELECT friend1 AS x`
	`FROM FRIENDS`
	`WHERE friend1 = friend2`

$friends(x, y) \wedge x = y$	`SELECT friend1 AS x friend2 AS y`
	`FROM FRIENDS`
	`WHERE friend1 = friend2`

$\exists x.friends(x, y)$	`SELECT friend2 AS y` `FROM FRIENDS`

Fun with Sentences

- All good people have friends.

  ∀x (Person(x) ∧ Good(x)) ⇒ ∃y(Friend(x,y))

- Some busy people have friends.

  ∃x Person(x) ∧ Busy(x) ∧ ∃y(Friend(x,y))

- Bad people have no friends.

  ∀x (Person(x) ∧ Bad(x)) ⇒ ¬ y(Friend(x,y))

  or equivalently: No bad people have friends.

  ¬∃x (Person(x) ∧ Bad(x) ∧ ∃y(Friend(x,y))

Fun with Sentences

- There is exactly one shoe.

  $\exists x\ Shoe(x) \wedge \forall y(Shoe(y) \Rightarrow (x=y))$

- There are exactly two shoes.

  $\exists x,y\ Shoe(x) \wedge Shoe(y) \wedge \neg(x=y) \wedge$
  $\forall z\ (Shoe(z) \Rightarrow (x=z) \vee (y=z))$

Fun with Sentences

- **Exercise:**
- **There is exactly even numbers of shoes.**

Axioms of Undireted Graph

- $\forall x \neg E(x,x)$ (antireflexivity)
- $\forall x \forall y (E(x,y) = E(y,x))$ (symmetry)

- E stands for edge relation of graph: E(x, y) is true if and only if there is an edge in the graph from the vertex that x denotes to the vertex that y denotes.

- Every node has exactly one edge leaving it: $\forall x \exists y$ $E(x, y) \wedge \forall x \forall y \forall z((E(x, y) \wedge E(x, z)) \Rightarrow y = z)$.

- Example: the friendship relation of Facebook

Example

"There is a vertex of degree at least three"

$\exists x \exists u \exists v \exists w \; u \neq v \wedge u \neq w \wedge v \neq w \wedge \mathrm{E}(x,u) \wedge \mathrm{E}(x,v) \wedge \mathrm{E}(x,w)$

Translation

- Try to translate the following sentence into first-order logic
- 美國國王是個禿頭

Russell's Theory of Descriptions

- $\exists x[(K(x) \wedge \forall y(K(y) \rightarrow x=y)) \wedge B(x)$
- 存在一個人，這人是美國國王且這人是唯一的且這人是禿頭

Validity and Satisfiability

- Validity problem (VAL): Given a formula A, is A valid?

- Satisfiability problem (SAT): Given a formula A, is A satisfiable?

- A is valid if M |= A for every model M.
- A is satisfiable if M |= A for some model M.
- Hence, A is valid iff ¬A is not satisfiable.

Example

- 假設:
- 任何哺乳類都有毛$\forall x(P(x) \rightarrow Q(x))$
- 任何有毛的動物可以被做成皮草$\forall x(Q(x) \rightarrow R(x))$
- 因此:
- 任何哺乳類動物皆可被做成皮草$\forall x(P(x) \rightarrow R(x))$
- $\forall x(P(x) \rightarrow Q(x))\ \forall x(Q(x) \rightarrow R(x)) \models \forall x(P(x) \rightarrow R(x))$

Example

- Valid:
- $\models \forall x (P(x) \vee \neg P(x))$
- $\models \forall x(B(x) \rightarrow M(x)) \rightarrow \forall x(\neg M(x) \rightarrow \neg B(x))$
- $\forall x(P(x) \rightarrow Q(x)) \; \forall x(Q(x) \rightarrow R(x)) \models \forall x(P(x) \rightarrow R(x))$

- Not valid but satisfiable:
- $\models \forall x (P(x))$

- unsatisfiable
- $\models \forall x (P(x) \wedge \neg P(x))$

Skolemization

- Every logician writes at least one book.
- $\forall x[Logic(x) \rightarrow \exists y[Book(y) \wedge Write(x, y)]]$
- Eliminate Implication:
- $\forall x[\neg Logic(x) \vee \exists y[Book(y) \wedge Write(x, y)]]$
- Skolemize: substitute y by g(x)
- $\forall x[\neg Logic(x) \vee [Book(g(x)) \wedge Write(x, g(x))]]$

Skolemization

- All students of a logician read one of their teacher's books.
- $\forall x \forall y[Logic(x) \wedge StudentOf(y, x) \rightarrow \exists z[Book(z) \wedge Write(x, z) \wedge Read(y, z)]]$

- Eliminate Implication:
- $\forall x \forall y[\neg Logic(x) \vee \neg StudentOf(y, x) \vee \exists z[Book(z) \wedge Write(x, z) \wedge Read(y, z)]]$

- Skolemize: substitute z by h(x, y)
- $\forall x \forall y[\neg Logic(x) \vee \neg StudentOf(y, x) \vee [Book(h(x, y)) \wedge Write(x, h(x, y)) \wedge Read(y, h(x, y))]]$

Drinker Paradox

- "There is someone in the pub such that, if he is drinking, then everyone in the pub is drinking."

- $\exists x\,(D(x) \rightarrow \forall y\,D(y))$
- Negation: $\neg(\exists x\,(D(x) \rightarrow \forall y\,D(y)))$
- $\neg(\exists x\,(\neg\, D(x) \vee \forall y\,D(y)))$
- $\forall x(\,\neg\,\neg D(x) \wedge \exists\, y\,\neg D(y))$
- $\forall x(\,D(x) \wedge \exists\, y\,\neg\,D(y))$
- $\forall x\,\exists\, y(\,D(x) \wedge \neg\,D(y))$
- $\forall x\,(\,D(x) \wedge \neg\,D(f(x)))$
- Contradiction!

Example: Euclidean Geometry

- Let B(x, y, z) means that y lies between x and z.

$\exists x \, \exists y \;\; x \neq y$

$\forall x \, \forall y \;\; \mathrm{B}(x, y, x) \rightarrow x = y$ *the one-point set is convex*

$\forall x \, \forall y \, \forall z \quad (x, y, z) \rightarrow \quad (z, y, x)$ *direction is irrelevant*

$\forall x \, \forall y \, \forall z \, \forall w \quad (x, y, z) \wedge \quad (x, z, w) \rightarrow \quad (y, z, w)$ *transitivity*

$\forall x \, \forall y \, \exists z \;\; y \neq z \wedge \quad (x, y, z)$ *extension axiom*

$\forall x \, \forall z \;\; x \neq z \rightarrow \exists y \, (x \neq y \wedge \; y \neq z \wedge \quad (x, y, z))$ *density*

$\forall x \, \forall y \, \forall z \quad (x, y, z) \vee \quad (y, z, x) \vee \quad (z, x, y)$ *dimension axiom*

Lemma

a. $\forall x \, \forall y \quad (x, x, y)$

b. $\forall x \, \forall y \quad (x, y, y)$

c. $\forall x \, \forall y \, \forall z \, (x = y \wedge y = z \rightarrow \quad (x, y, z))$

d. $\forall x \, \forall y \, \forall z \quad (x, y, z) \wedge \quad (x, z, y) \rightarrow \; y = z$

e. $\forall x \, \forall y \, \forall z \quad (x, y, z) \wedge \quad (y, x, z) \rightarrow \; x = y$

f. $\forall x \, \forall y \, \forall z \, \forall w \quad (x, y, w) \wedge \quad (y, z, w) \rightarrow \quad (x, y, z)$

g. $\forall x \, \forall y \, \forall z \, \forall w \quad (x, y, z) \wedge \quad (x, z, w) \rightarrow \quad (x, y, w)$

h. $\forall x \, \forall y \, \forall z \, \forall w \quad (x, y, w) \wedge \quad (y, z, w) \rightarrow \quad (x, z, w)$

益智問題

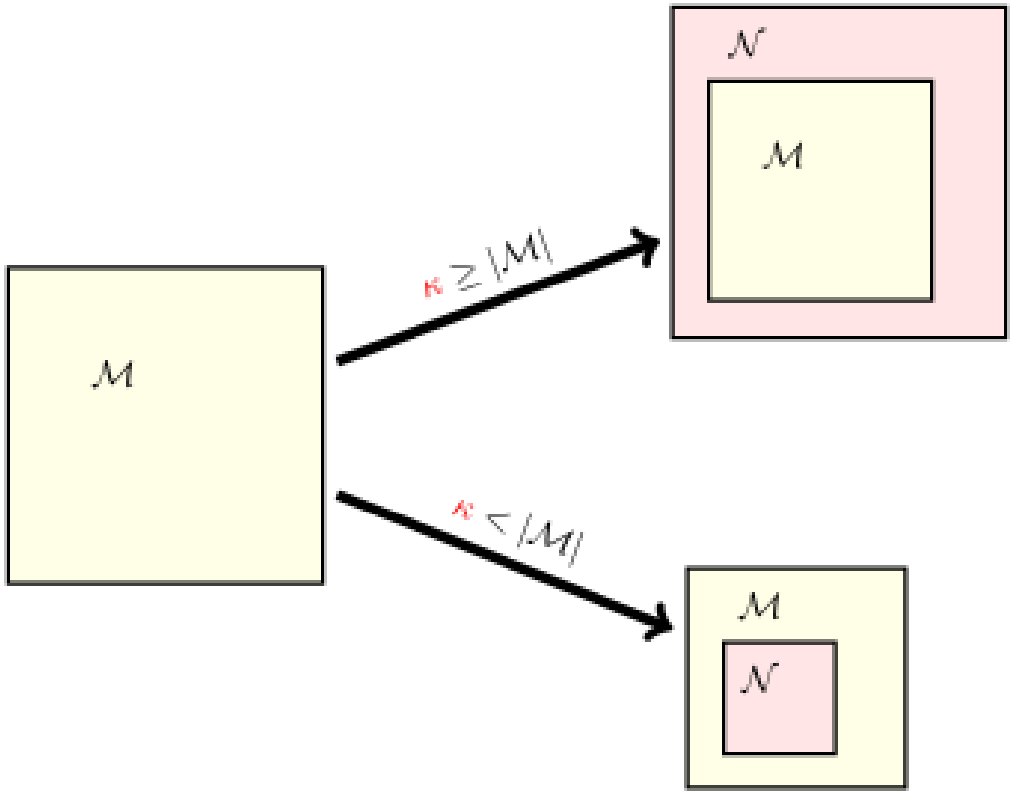
- 有四對夫妻一起參加一個聖誕舞會。丈夫的名字是吉米、彼得、約翰和保羅，而妻子的名子是瑪麗、珍妮、安麗和溫蒂。現在已知：

- 1. 吉米的妻子不是與自己的丈夫在跳舞，而是與瑪莉的丈夫跳舞
- 2. 保羅和溫蒂不在跳舞
- 3. 彼得在彈吉他，而安麗在彈鋼琴
- 4. 溫蒂的丈夫不是彼得

- 請問四位丈夫的妻子各是誰？

Löwenheim–Skolem Thereom

- If a set of sentences has an infinite model, then for every infinite cardinal number κ it has a model of size κ.
- If a set of sentences S has any finite , then S has a model with infinitely many elements.



Compcatness Theorem

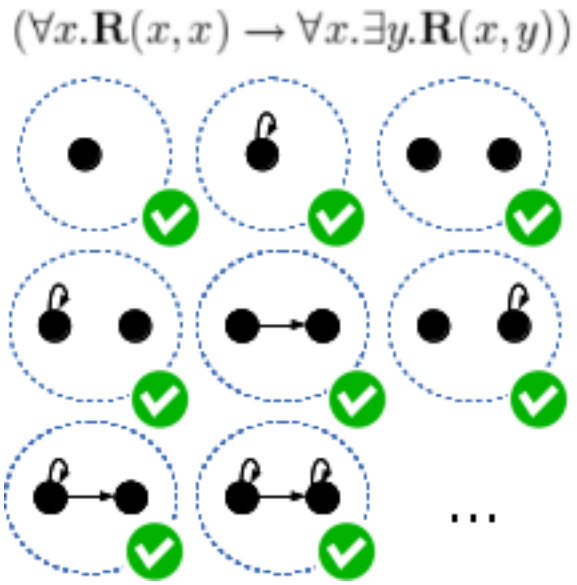
- Let S be a set of wffs of first-order logic. If every finite subset of S is satisfiable, then S is satisfiable.

- Corollaries:
- If a set S is unsatisfiable then some finite subset of S is unsatisfiable.

- If S |= A then there is a finite subset of S, Σ , such that Σ |= A

Gödel's completeness theorem

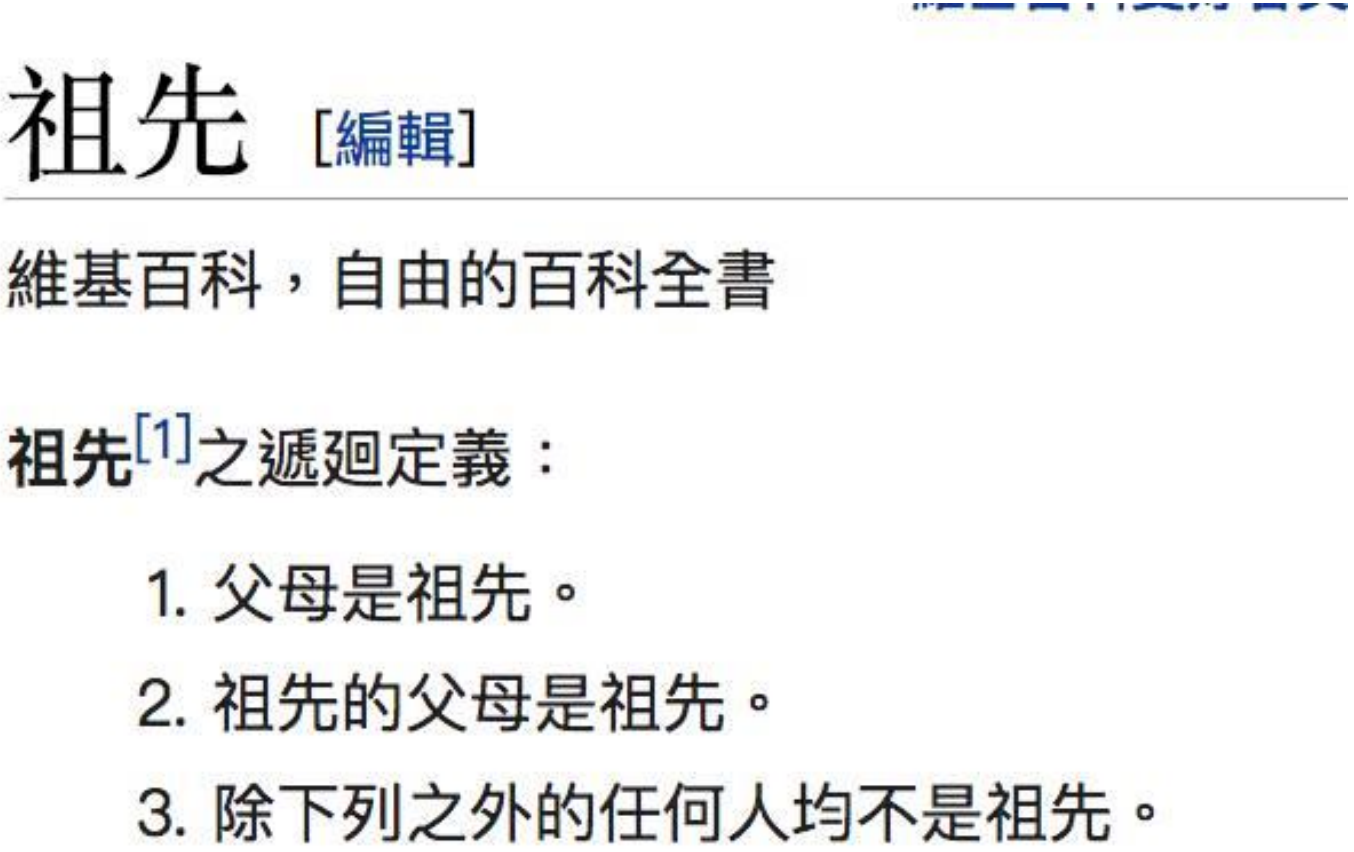
- If a first order formula is logically valid then there is a finite deduction (a formal proof) of the formula.



$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x.\mathbf{R}(x,x)}{\mathbf{R}(x,x)}(E\forall)}{\exists y.\mathbf{R}(x,y)}(I\exists)}{\forall x.\exists y.\mathbf{R}(x,y)}(I\forall)}{(\forall x.\mathbf{R}(x,x) \rightarrow \forall x.\exists y.\mathbf{R}(x,y))}(I\rightarrow)(1)}\ (1)$$

Fun with Sentences

- How to define the ancestral relation in first order logic?
- How to define the relation of reachability?



祖先 [編輯]

維基百科，自由的百科全書

祖先[1]之遞迴定義：

1. 父母是祖先。
2. 祖先的父母是祖先。
3. 除下列之外的任何人均不是祖先。

Languages for defining new relations

- First-order logic predicate calculus
  $\forall x, y\ path(x, y) \Leftrightarrow edge(x, y) \vee \exists z(edge(x, z) \wedge path(z, y))$
- Prolog/CLP($\mathcal{X}$) clauses

```
path(X,Y):- edge(X,Y).
path(X,Y):- edge(X,Z), path(Z,Y).
```

- Concurrent constraint process languages CC($\mathcal{X}$)
  Process $A = c \mid p(x) \mid (A \parallel A) \mid A + A \mid ask(c) \rightarrow A \mid \exists x A$
  $path(X, Y) :: edge(X, Y) + \exists Z(edge(X, Z) || path(Z, Y))$
- Constraint libraries in object-oriented/functional/imperative languages (ILOG, Koalog, etc.).

Languages for defining new relations

- First-order logic predicate calculus
  $\forall x, y\ path(x, y) \Leftrightarrow edge(x, y) \vee \exists z(edge(x, z) \wedge path(z, y))$
- Prolog/CLP($\mathcal{X}$) clauses

```
path(X,Y):- edge(X,Y).
path(X,Y):- edge(X,Z), path(Z,Y).
```

- Concurrent constraint process languages CC($\mathcal{X}$)
  Process $A = c \mid p(x) \mid (A \parallel A) \mid A + A \mid ask(c) \rightarrow A \mid \exists x A$
  $path(X, Y) :: edge(X, Y) + \exists Z(edge(X, Z) || path(Z, Y))$
- Constraint libraries in object-oriented/functional/imperative languages (ILOG, Koalog, etc.).

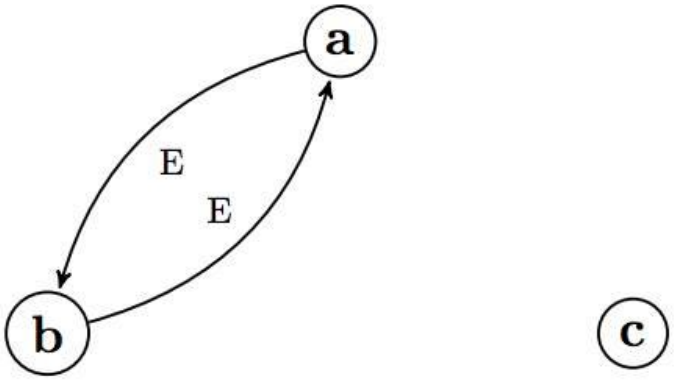


Figure 1: directed graph (Predicate $E(x, y)$)

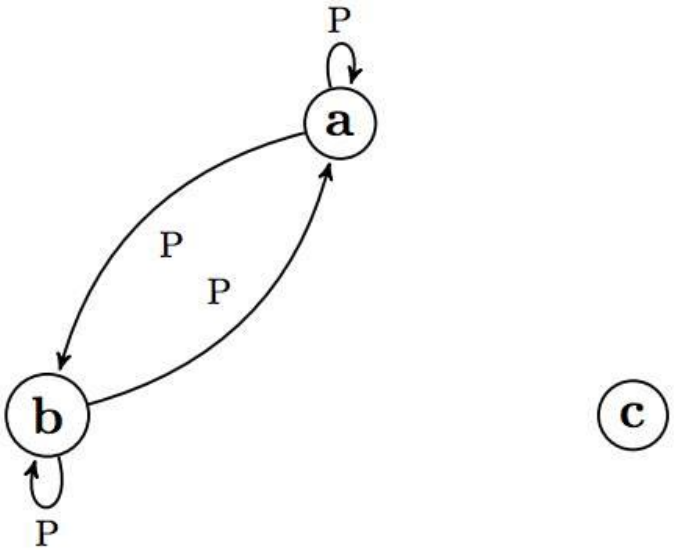


Figure 2: It's transitive closure (Predicate $Path(x, y)$)

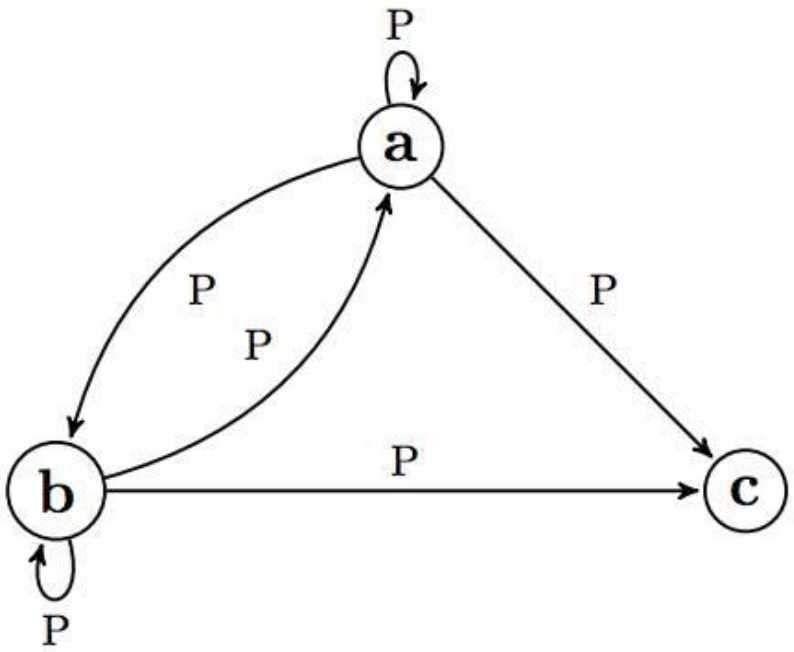


Figure 3: A model of $\forall x \forall y(Path(x, y) \leftrightarrow (Edge(x, y) \vee \exists z(Edge(x, z) \wedge Path(z, y))))$

$\forall x, y\ path(x, y) \Leftrightarrow edge(x, y) \vee \exists z(edge(x, z) \wedge path(z, y))$

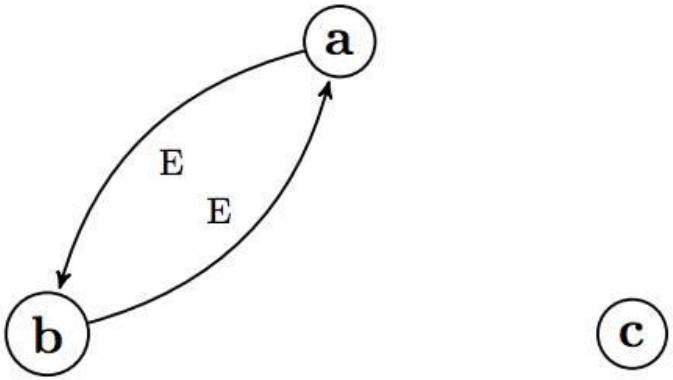


Figure 1: directed graph (Predicate $E(x, y)$)

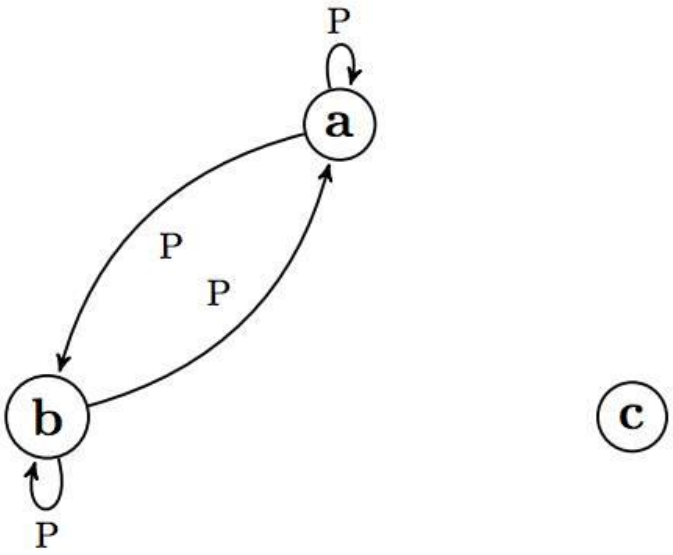


Figure 2: It's transitive closure (Predicate $Path(x, y)$)

The Structure of the Natural Numbers

0 ← there's a first one

s(0) ← then exactly one next one

s(s(0)) ← then exactly one next one

s(s(s(0)))

ssss0

sssss0

ssssss0

and so on…

Peano Axioms

The principles of the Peano arithmetic are as follows.

- 1. Zero is a natural number.
- 2. If *x* is a number, the successor of *x* is a number as well.
- 3. Zero is not the successor of any number.
- 4. Two numbers of which the successors are equal are themselves equal.
- 5. (induction axiom.) If a set *s* of numbers contains zero and also the successor of every number in *s*, then every number is in *s*.

First Order Peano Arithmetic

- PA1: $\forall x\ s(x) \neq 0$
- PA2: $\forall x\ \forall y\ (s(x) = s(y) \rightarrow x = y)$
- PA3: $\forall x\ x + 0 = x$
- PA4: $\forall x\ \forall y\ x + s(y) = s(x + y)$
- PA5: $\forall x\ \ x * 0 = 0$
- PA6: $\forall x\ \forall y\ \ x * s(y) = (x * y) + x$
- Induction: $(\phi(0) \wedge \forall x\ (\phi(x) \rightarrow \phi(s(x)))) \rightarrow \forall x\ \phi(x)$

Theorem of First Order Peano Theory

- Exercise:
- (commutativity of addition):
- For any $x, y \in N$, $x + y = y + x$.
- (Cancellative law)
- For any $x, y, z \in N$, if $x + z = y + z$, then $x = y$.
- (Distributive law)
- If $x, y, z \in N$ then $x \cdot (y + z) = x \cdot y + x \cdot z$ and $(y + z) \cdot x = y \cdot x + z \cdot x$.
- (Associative law for multiplication)
- For any $x, y, z \in N$, $x(yz) = (xy)z$.

Sentence for Fun!

- Prime(x) is true if x is prime:
- $\forall y \forall z((x = y \times z) \Rightarrow ((y = 1) \vee (y = x)))$

- Goldbach's conjecture:
- Every even integer greater than 2 can be expressed as the sum of two primes.
- $\forall x(\exists y(y > 1 \wedge x = y + y) \Rightarrow \exists z_1 \exists z_2(\text{Prime}(z_1) \wedge \text{Prime}(z_2) \wedge x = z_1 + z_2))$

Gödel's incompleteness theorems

- No consistent system of axioms whose theorems can be listed by an algorithm is capable of proving all truths about the arithmetic of the natural numbers.

- For any such formal system, the system cannot demonstrate its own consistency.

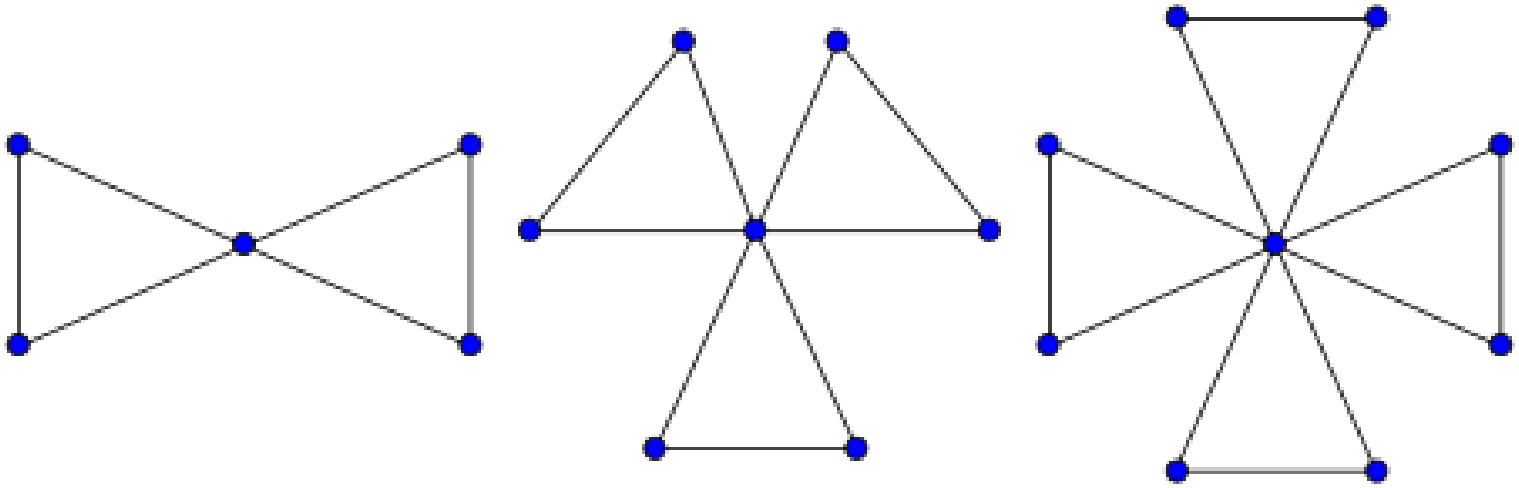
- FO-SAT = {ϕ : ϕ is a satisfiable sentence of first-order logic}.
- FO-Valid = {ϕ : ϕ is a valid sentence of first-order logic}.

Theorem FO-SAT is undecidable!

Corollary FO-Valid is undecidable!

THE FRIENDSHIP THEOREM

- Suppose in a group of people we have the situation that any pair of persons has precisely one common friend. Then there is always a person (the “politician”) who is everybody’s friend.



Formalization

- $\forall x \neg xEx$ (antireflexivity)
- $\forall x \forall y(E(x,y) = E(y,x))$ (symmetry)
- $\forall x\ \forall y\ (\ x!=y \rightarrow \exists z(E(x,z) \land E(y,z) \land \forall w(E(x,w) \land E(y,w) \rightarrow w=z)))$

Surprise ！ ！ ！！

- Finiteness Is Not First-order Definable ！ ！

Theorem There is no first-order formula ϕ such that $M \models \phi$ iff the domain of M is finite.

Trakhtenbrot's theorem

Definition A sentence φ, is called *finitely satisfiable* if there exists a finite model M such that M |= φ

Theorem no algorithm exists that always correctly decides whether a FO sentence φ is finitely satisfiable.

Satisfiable:

$\exists x.\exists y.\forall z.(R(x,z) \rightarrow R(y,z))$

$\exists x.\exists y.T(x) \vee \exists z.S(x,z)$

Unsatisfiable:

$\forall x.\forall y.\forall z.(R(x,y) \wedge R(x,z) \rightarrow y=z)$

$\wedge \exists y.\ \forall x.\ \text{not}\ R(x,y)$

- Consequently, the problem of validity in first-order logic on the class of all finite models is undecidable.

Second Order Logic

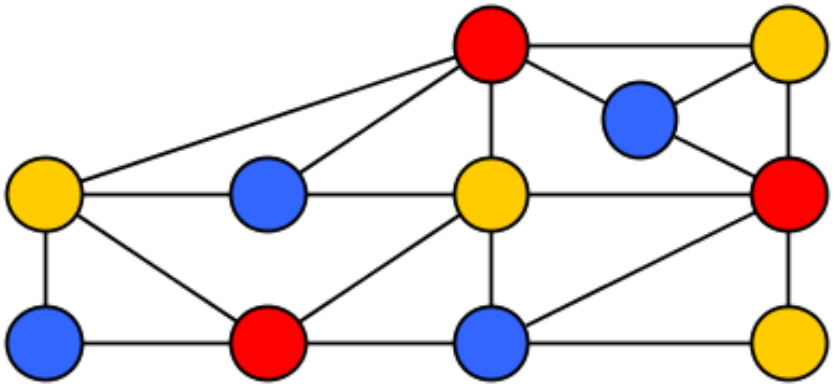
- Finiteness:
- $\forall F(\forall x \forall y\ (x \neq y \rightarrow F(x) \neq F(y)) \rightarrow \forall y \exists x\ (y = F(x)))$
- Even:
- $\exists F \forall x[F(x) \neq x \wedge F(F(x)) = x]$

- the above sentence is true in a structure iff every injective function f is a bijection i.e. the domain is a finite set. This proofs that in SO logic we can characterize the finiteness of the universe.

Second Order Logic

"The vertices can be colored red, green, blue such that the endpoints of each edge receive different colors"

$$\exists R \exists G \exists B: (\forall x: (R(x) \vee G(x) \vee B(x)) \wedge$$
$$(\forall x \forall y: E(x,y) \rightarrow \neg((R(x) \wedge R(y)) \vee (G(x) \wedge G(y)) \vee (B(x) \vee B(y)))))$$



Theory of first order logic

- Presburger arithmetic: $\sum$ = {0,1, ‘+’, ‘=’}
- Peano arithmetic: $\sum$ = {0,1, ‘+’, ‘*’, ‘=’}
- Theory of reals
- Theory of integers
- Theory of arrays
- Theory of pointers
- Theory of sets
- Theory of recursive data structures
- …

Satisfiability Modulo Theories

- “satisfiability modulo theories (SMT) problem is a decision problem for logical formulas with respect to combinations of background theories expressed in classical first-order logic with equality.”

Platform			Features					
Name	OS	License	SMT-LIB	CVC	DIMACS	Built-in theories	API	SMT-COMP [2]
ABsolver	Linux	CPL	v1.2	No	Yes	linear arithmetic, non-linear arithmetic	C++	no
Alt-Ergo	Linux, Mac OS, Windows	CeCILL-C (roughly equivalent to LGPL)	partial v1.2 and v2.0	No	No	empty theory, linear integer and rational arithmetic, non-linear arithmetic, polymorphic arrays, enumerated datatypes, AC symbols, bitvectors, record datatypes, quantifiers	OCaml	2008
Barcelogic	Linux	Proprietary	v1.2			empty theory, difference logic	C++	2009
Beaver	Linux, Windows	BSD	v1.2	No	No	bitvectors	OCaml	2009
Boolector	Linux	GPLv3	v1.2	No	No	bitvectors, arrays	C	2009
CVC3	Linux	BSD	v1.2	Yes		empty theory, linear arithmetic, arrays, tuples, types, records, bitvectors, quantifiers	C/C++	2010
CVC4	Linux, Mac OS, Windows	BSD	Yes	Yes		rational and integer linear arithmetic, arrays, tuples, records, inductive data types, bitvectors, strings, and equality over uninterpreted function symbols	C++	2010
Decision Procedure Toolkit (DPT)	Linux	Apache	No				OCaml	no
iSAT	Linux	Proprietary	No			non-linear arithmetic		no
MathSAT	Linux, Mac OS, Windows	Proprietary	Yes		Yes	empty theory, linear arithmetic, bitvectors, arrays	C/C++, Python, Java	2010
MiniSmt	Linux	LGPL	partial v2.0			non-linear arithmetic		2010
Norn								
OpenCog	Linux	AGPL	No	No	No	probabilistic logic, arithmetic. relational models	C++, Scheme, Python	no
OpenSMT	Linux, Mac OS, Windows	GPLv3	partial v2.0		Yes	empty theory, differences, linear arithmetic, bitvectors	C++	2011
raSAT	Linux	GPLv3	v2.0			real and integer nonlinear arithmetic		2014, 2015
SatEEn	?	Proprietary	v1.2			linear arithmetic, difference logic	none	2009
SMTInterpol	Linux, Mac OS, Windows	LGPLv3	v2.0			uninterpreted functions, linear real arithmetic, and linear integer arithmetic	Java	2012
SMCHR	Linux, Mac OS, Windows	GPLv3	No	No	No	linear arithmetic, nonlinear arithmetic, heaps	C	no
SMT-RAT	Linux, Mac OS	MIT	v2.0	No	No	linear arithmetic, nonlinear arithmetic	C++	2015
SONOLAR	Linux, Windows	Proprietary	partial v2.0			bitvectors	C	2010
Spear	Linux, Mac OS, Windows	Proprietary	v1.2			bitvectors		2008
STP	Linux, OpenBSD, Windows, Mac OS	MIT	partial v2.0	Yes	No	bitvectors, arrays	C, C++, Python, OCaml, Java	2011
SWORD	Linux	Proprietary	v1.2			bitvectors		2009
UCLID	Linux	BSD	No	No	No	empty theory, linear arithmetic, bitvectors, and constrained lambda (arrays, memories, cache, etc.)		no
veriT	Linux, OS X	BSD	partial v2.0			empty theory, rational and integer linear arithmetics, quantifiers, and equality over uninterpreted function symbols	C/C++	2010

Example

```
from z3 import *

x = Real('x')
y = Real('y')
s = Solver()
s.add(x + y > 5, x > 1, y > 1)
print(s.check())
print(s.model())
```

Example:Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Tarski's exponential function problem

- Whether the theory of the real numbers together with the exponential function is decidable

- Delta-Decision Procedures

Delta-Decision Procedures

Delta-Complete Decision Procedures for Satisfiability over the Reals

Sicun Gao, Jeremy Avigad, Edmund Clarke

(Submitted on 16 Apr 2012 (v1), last revised 17 Sep 2012 (this version, v2))

We introduce the notion of "\delta-complete decision procedures" for solving SMT problems over the real numbers, with the aim of handling a wide range of nonlinear functions including transcendental functions and solutions of Lipschitz-continuous ODEs. Given an SMT problem \varphi and a positive rational number \delta, a \delta-complete decision procedure determines either that \varphi is unsatisfiable, or that the "\delta-weakening" of \varphi is satisfiable. Here, the \delta-weakening of \varphi is a variant of \varphi that allows \delta-bounded numerical perturbations on \varphi. We prove the existence of \delta-complete decision procedures for bounded SMT over reals with functions mentioned above. For functions in Type 2 complexity class C, under mild assumptions, the bounded \delta-SMT problem is in NP^C. \delta-Complete decision procedures can exploit scalable numerical methods for handling nonlinearity, and we propose to use this notion as an ideal requirement for numerically-driven decision procedures. As a concrete example, we formally analyze the DPLL<ICP> framework, which integrates Interval Constraint Propagation (ICP) in DPLL(T), and establish necessary

Delta-Weakening and Pertubations

Definition (**δ-Weakening and Perturbations**). *Let $\delta \in \mathbb{Q}^+ \cup \{0\}$ be a constant and φ be a Σ_1-sentence in standard form:*

$$\varphi := \exists^{\boldsymbol{I}} \boldsymbol{x}. \bigwedge_{i=1}^{m} (\bigvee_{j=1}^{k_i} f_{ij}(\boldsymbol{x}) = 0).$$

The δ-weakening of φ defined as:

$$\varphi^{\delta} := \exists^{\boldsymbol{I}} \boldsymbol{x}. \bigwedge_{i=1}^{m} (\bigvee_{j=1}^{k} |f_{ij}(\boldsymbol{x})| \leq \delta).$$

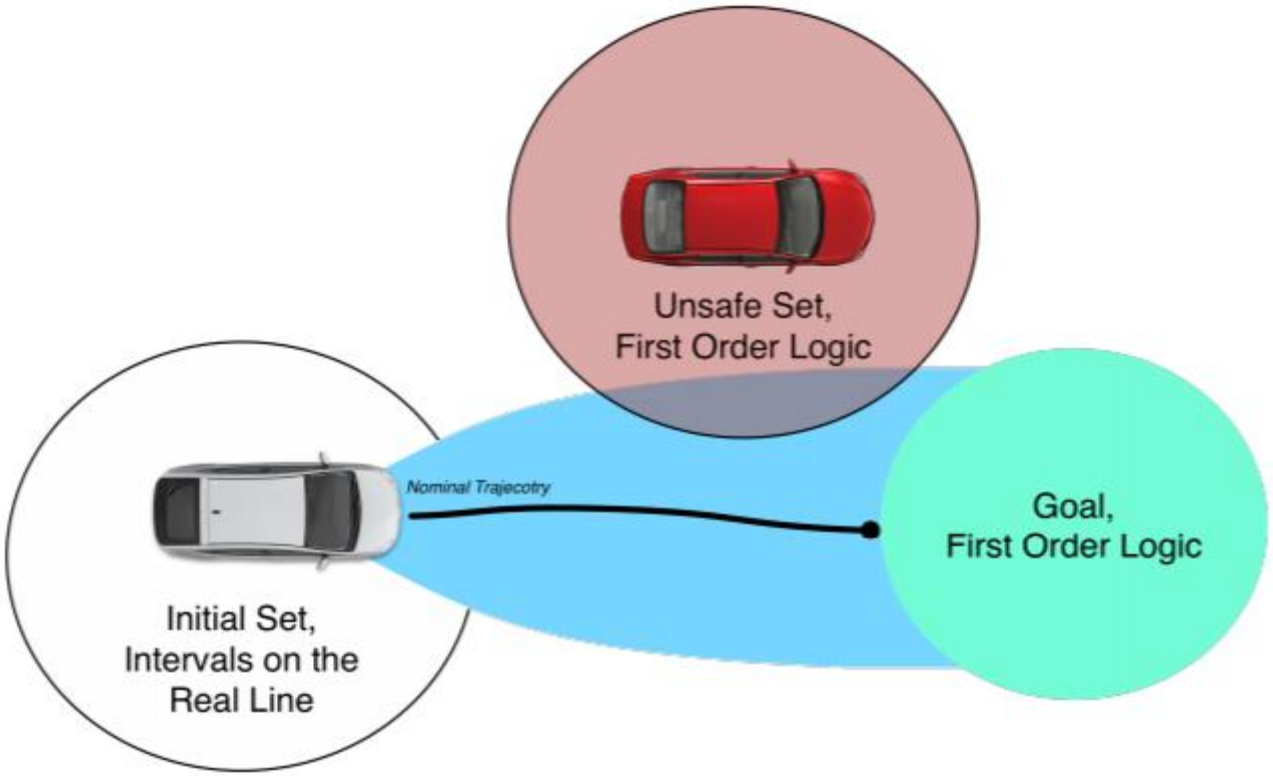
Also, a δ-perturbation is a constant vector $\boldsymbol{c} = (c_{11}, ..., c_{mk_m})$, $c_{ij} \in \mathbb{Q}$, satisfying $||\boldsymbol{c}|| \leq \delta$, such that the $\boldsymbol{c}$-perturbed form of φ is given by:

$$\varphi^{\boldsymbol{c}} := \exists^{\boldsymbol{I}} \boldsymbol{x}. \bigwedge_{i=1}^{m} (\bigvee_{j=1}^{k} f_{ij}(\boldsymbol{x}) = c_{ij}).$$

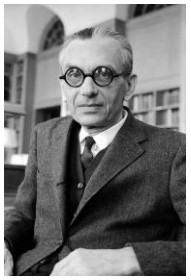
Example

$$\exists^{[3.0,3.14]} x_1 . \exists^{[-7.0,5.0]} x_2 . 2 \times 3.14159265 - 2x_1 \arcsin\left(\cos 0.797 \times \sin\left(\frac{3.14159265}{x_1}\right)\right)$$

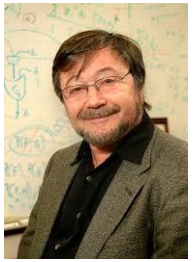
$$\leq -0.591 - 0.0331x_2 + 0.506 + 1.0$$


Unsafe Set,
First Order Logic
Nominal Trajecotry
Goal,
First Order Logic
Initial Set,
Intervals on the
Real Line

Logic + Probability = Probabilistic Logic aka Statistical Relational Learning



Logic



Probabilities

Add Probabilities

Add Relations

Probabilistic Logic