Finite Model Finding in Satisfiability Modulo Theories

Andrew Reynolds

December 3, 2013

University of Iowa

Motivation

- Many aspects of modern life are dependent upon software
- Correctness of software is often highly critical
  - Flight control, Bank records, Medical Devices, …
- Growing need for *automated reasoning*
  - For software verification and other applications

Approaches to Automated Reasoning

- Boolean Satisfiability Solvers
  - Fast, Handle Decidable Logic
  - Cons : May be difficult to encode problem into SAT
- Automated First-Order Theorem Provers
  - Handle problems in an expressive natural encoding
  - Cons : Logic can be Undecidable
- *Alternative* : Satisfiability Modulo Theories (SMT)
  - Incorporate specialized procedures for theories
    - Arithmetic, bitvectors, arrays, datatypes, …
  - Many problems can be expressed as SMT problems

SMT Solvers

- SMT solvers are powerful tools that
  - Are used in many formal methods applications
  - Have optimized performance due to combination of:
    - Off-the-shelf SAT solver
    - Fast decision procedures for (ground) constraints
  - May generate:
    - Proofs
      - Theorem proving, software/hardware verification
    - Models
      - Failing instances of aforementioned applications
      - Invariant synthesis, scheduling, test case generation

SMT: Limitations

- Ongoing challenge: *quantified* formulas
  - Are useful for:
    - Frame axioms in software verification
    - Universal safety properties
    - Axiomatization of unsupported theories
    - …
  - Needed by a growing number of SMT-based applications
- Current methods for handling quantifiers in SMT:
  - Heuristic methods for answering “UNSAT”
  - Limited capability of answering “SAT”
    - Often will return “UNKNOWN” after some effort

Contributions

- Finite Model Finding in SMT
  - New approach for handling quantifiers in SMT
  - Different from ATP finite model finders:
    - Native support for background theories
  - Different from SMT solvers:
    - Increased ability to answer "satisfiable"

Outline

- Intro to Satisfiability Modulo Theories (SMT)
- Finite Model Finding in SMT
  - Details of Approach
  - Theoretical Properties
  - Experimental Results
- Extension to Bounded Integer Quantification

Satisfiability Modulo Theories

$(f(a) = b \vee f(a) = c) \wedge c+1 = b \wedge f(c) = g(c)$

Satisfiability Modulo Theories

$$(f(a) = b \lor f(a) = c) \land c+1 = b \land f(c) = g(c)$$

$\Downarrow$ Abstract to propositional logic

$$(A \lor B) \land C \land D$$

Satisfiability Modulo Theories

$$(f(a) = b \lor f(a) = c) \land c+1 = b \land f(c) = g(c)$$

$$(A \lor B) \land C \land D$$

$\underbrace{A}_{\texttt{true}}$, $\underbrace{C}_{\texttt{true}}$, $\underbrace{D}_{\texttt{true}}$

Find satisfying assignment: A , C , D

Satisfiability Modulo Theories

$(f(a) = b \vee f(a) = c) \wedge c+1 = b \wedge f(c) = g(c)$

$(A \vee B) \wedge C \wedge D$

`true` (A), `true` (C), `true` (D)

Find satisfying assignment: A, C, D

Check T-consistency: $f(a) = b$, $c+1 = b$, $f(c) = g(c)$

$\Rightarrow$ This can be done with ground theory solver

SMT with Quantified Formulas

(f(a) = b ∨ f(a) = c) ∧ c+1 = b ∧ ∀x. f(x) = g(x)

(A ∨ B) ∧ C ∧ D

`true` `true` `true`

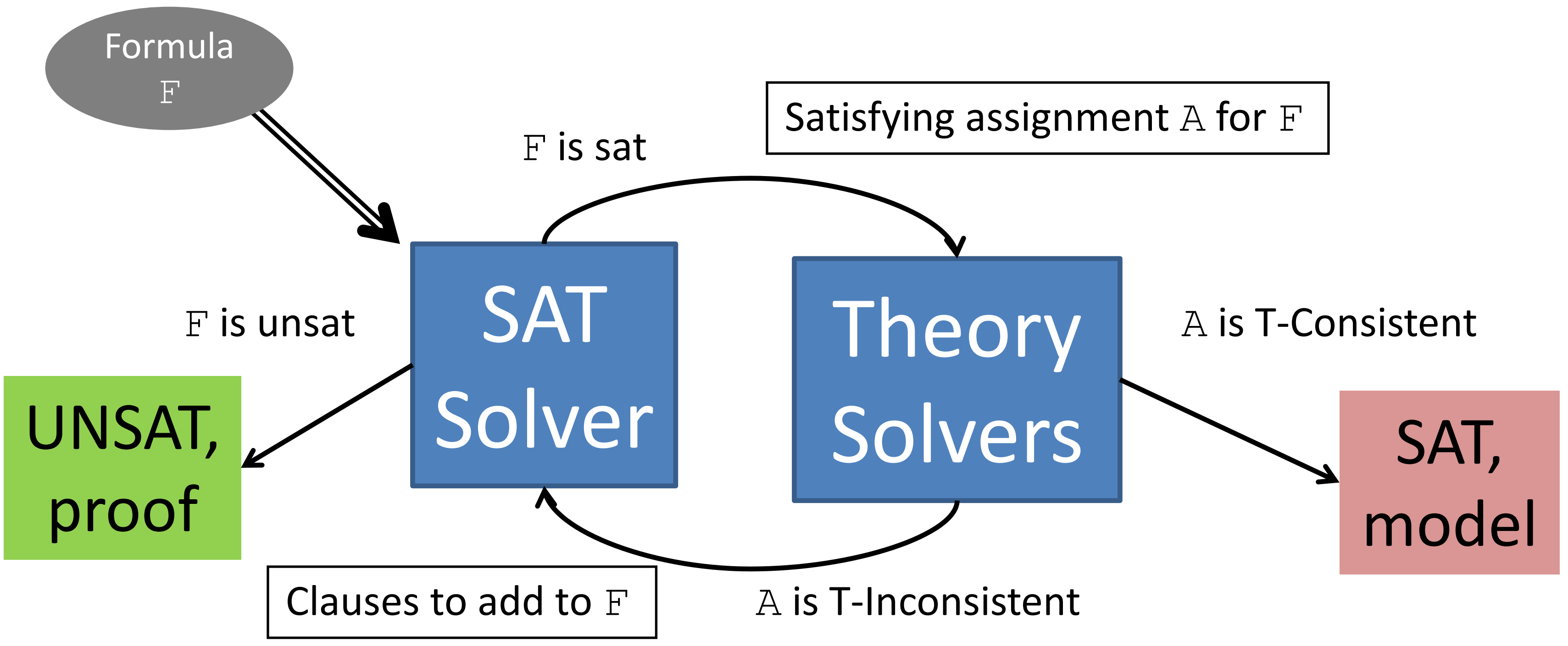
Find satisfying assignment: A , C , D

Check T-consistency: f(a) = b , c+1 = b , ∀x. f(x) = g(x)

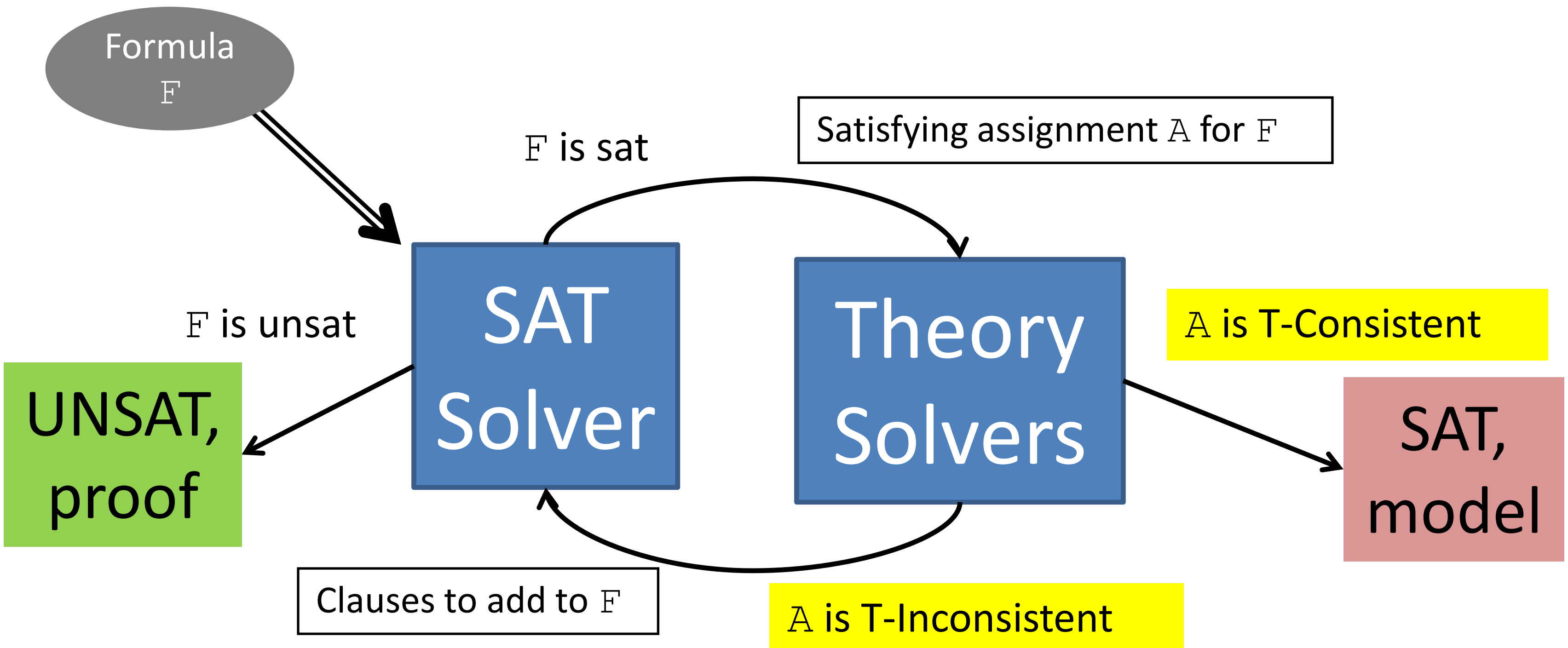
- Satisfying assignment contains *quantified formulas*

⇒ Challenge: This is generally undecidable

DPLL(T) Architecture

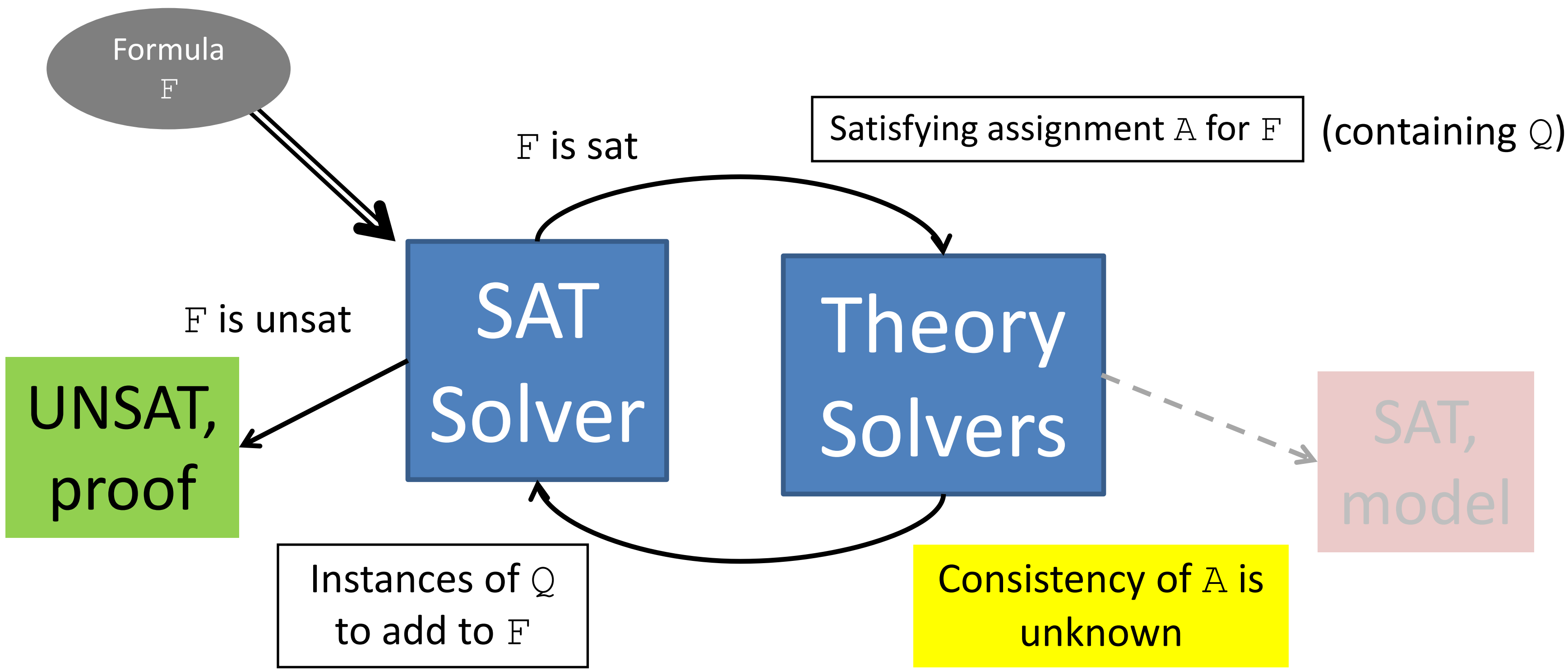


DPLL(T) Architecture : Challenge



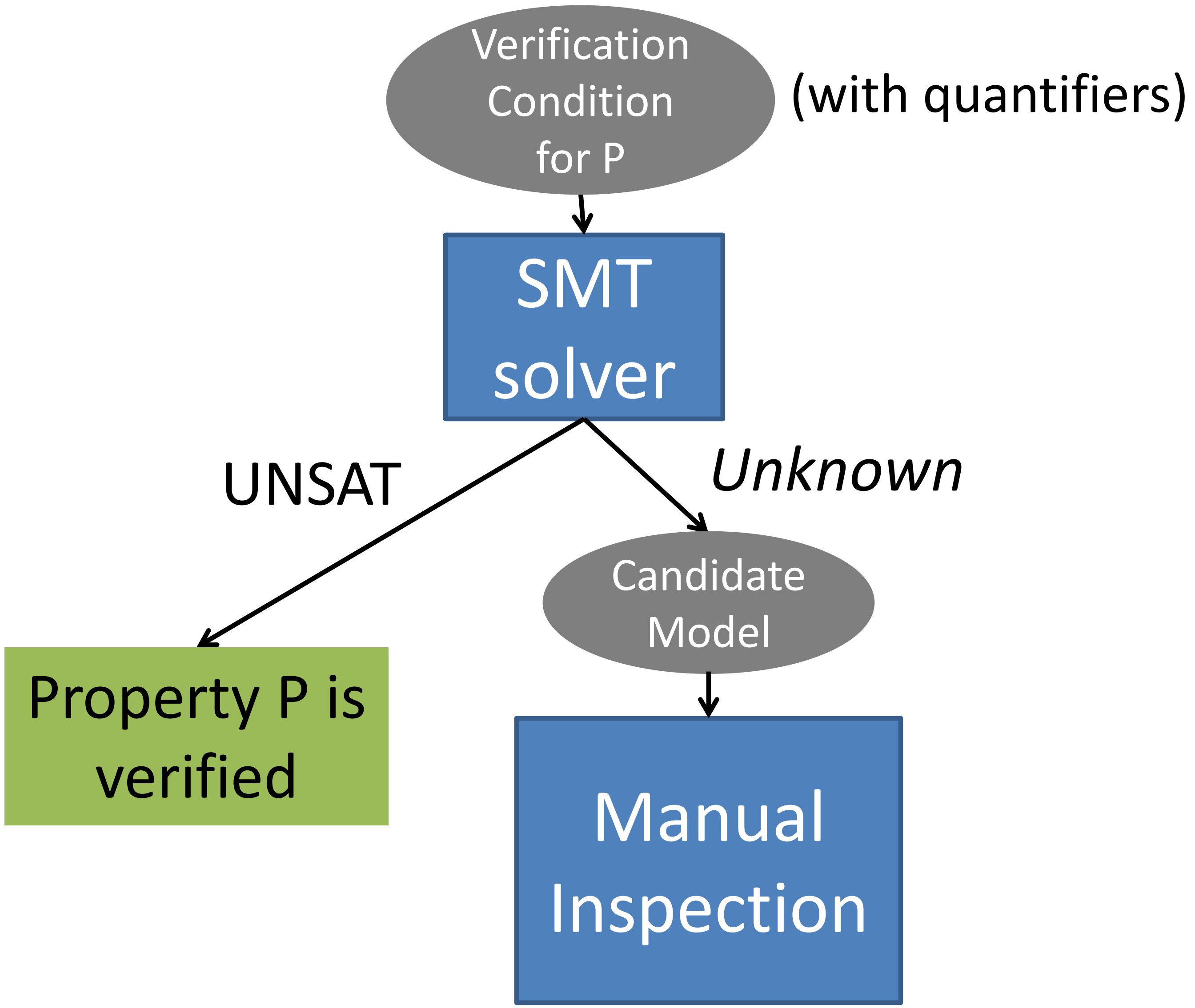
- Challenge: What if determining the consistency of A is difficult?
- For quantified formulas, determining T-consistency is *undecidable*

Heuristic Instantiation

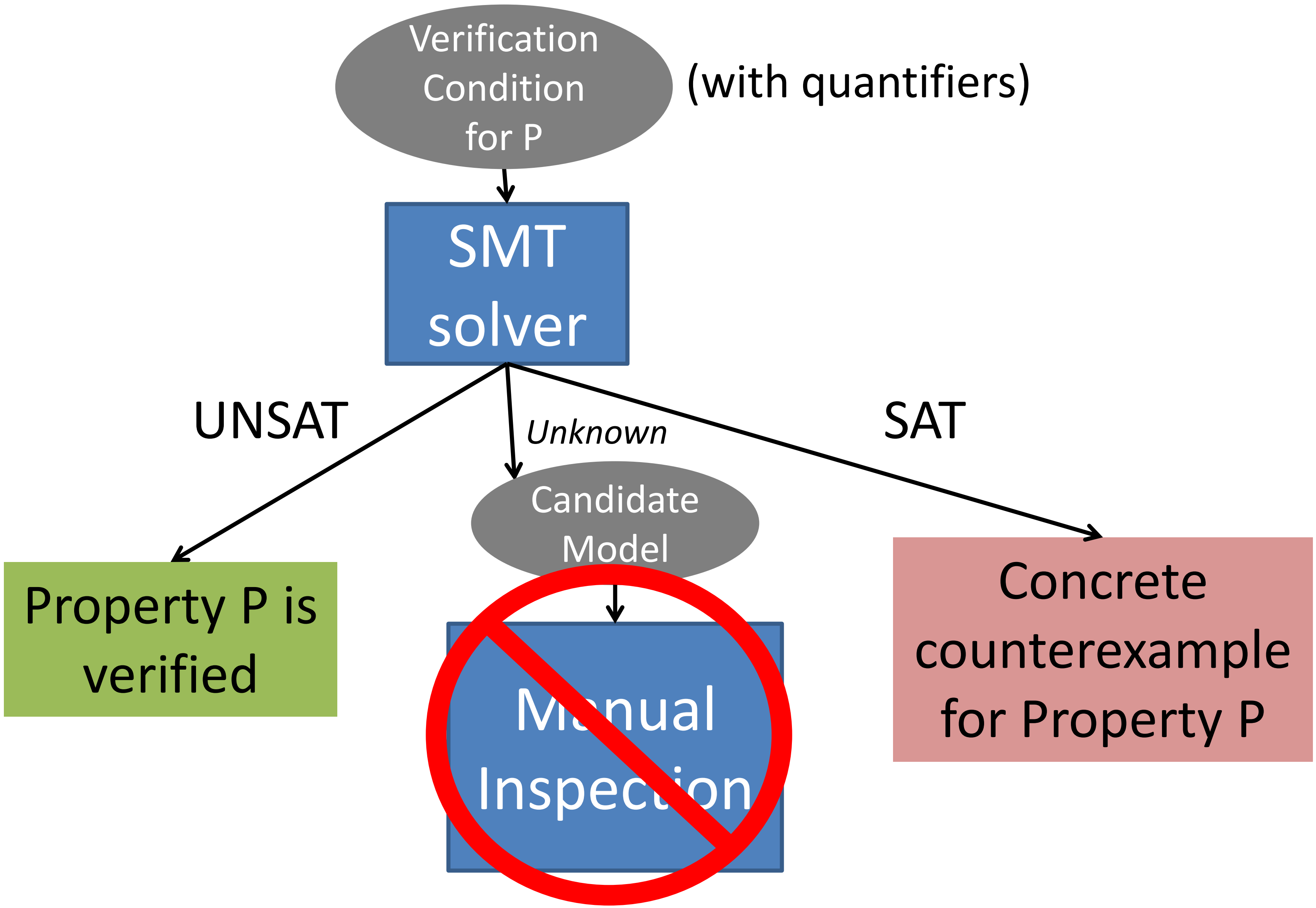


- If sat assignment contains quantified formula Q,
  - Heuristically add instances of Q to F [Detlefs et al 2003]
    - Typically based on pattern matching
    - May discover refutation, if right instances are added
    - *No way to answer SAT*

Why Models are Important



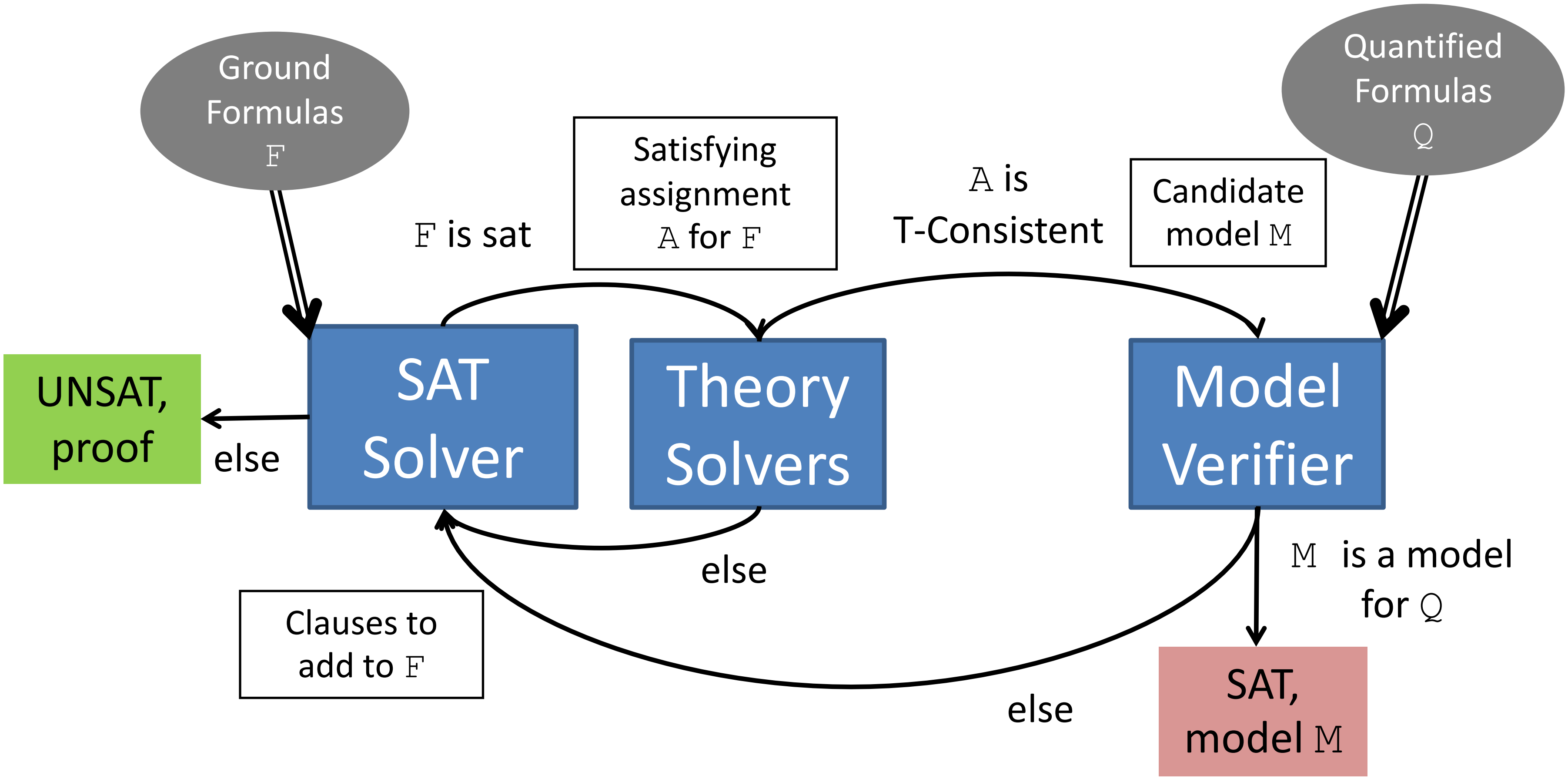
Why Models are Important



Model-Based Approach for Quantifiers

- Given:
  - Set of ground formulas $\mathtt{F}$
  - Set of universally quantified formulas $\mathtt{Q}$
- To determine the satisfiability of $\mathtt{F} \wedge \mathtt{Q}$,
  - Construct candidate models for $\mathtt{Q}$, based on satisfying assignments for $\mathtt{F}$
    - Model-Based Quantifier Instantiation (MBQI)
      - [Ge/deMoura 2009]

DPLL(T) Architecture (Extended)



When can we represent/check models for $\mathcal{Q}$?

- Focus of thesis: Finite Model Finding
  - Limited to quantifiers over:
    - Uninterpreted sorts
      - Can represent memory addresses, values, sets, etc.
    - Other finite sorts
      - Fixed width bitvectors, datatypes, …
- Useful in applications:
  - Software verification, automated theorem proving

Running Example

$person_1, person_2, person_3$: Person

NewYork, Boston, Seattle : City

salesman: Person $\rightarrow$ Bool

travels : Person $\times$ City $\rightarrow$ Bool

$\mathbb{F}$

distinct(NewYork, Boston, Seattle)

travels($person_1$, Boston)

$\neg$salesman($person_2$)

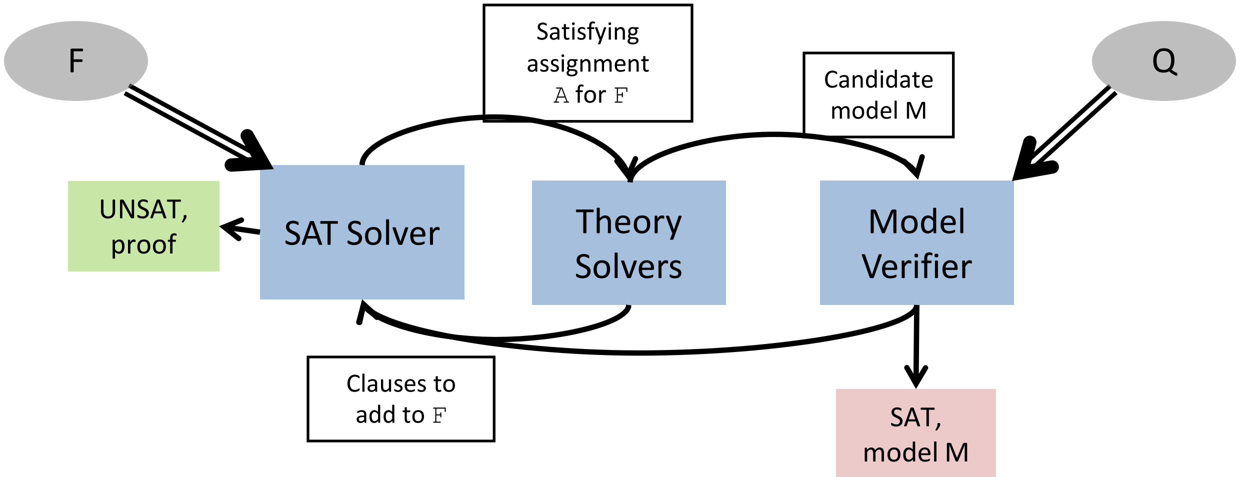
salesman($person_3$)

$\mathbb{Q}$

$\forall$ x : Person, y : City.

salesman(x) $\Rightarrow$ travels(x,y)

Running Example



F:

distinct(NewYork, Boston, Seattle)

travels($person_1$, Boston)

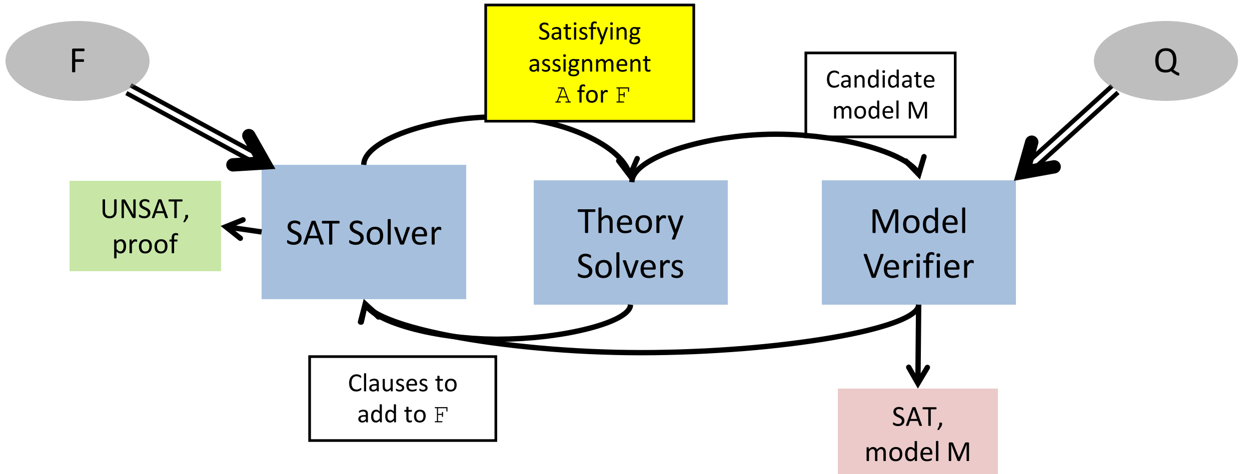
$\neg$salesman($person_2$)

salesman($person_3$)

Q:

$\forall$ x y . salesman(x) $\Rightarrow$ travels(x,y)

Find Satisfying Assignment A for F



true — distinct(NewYork, Boston, Seattle)

true — travels(person$_1$, Boston)

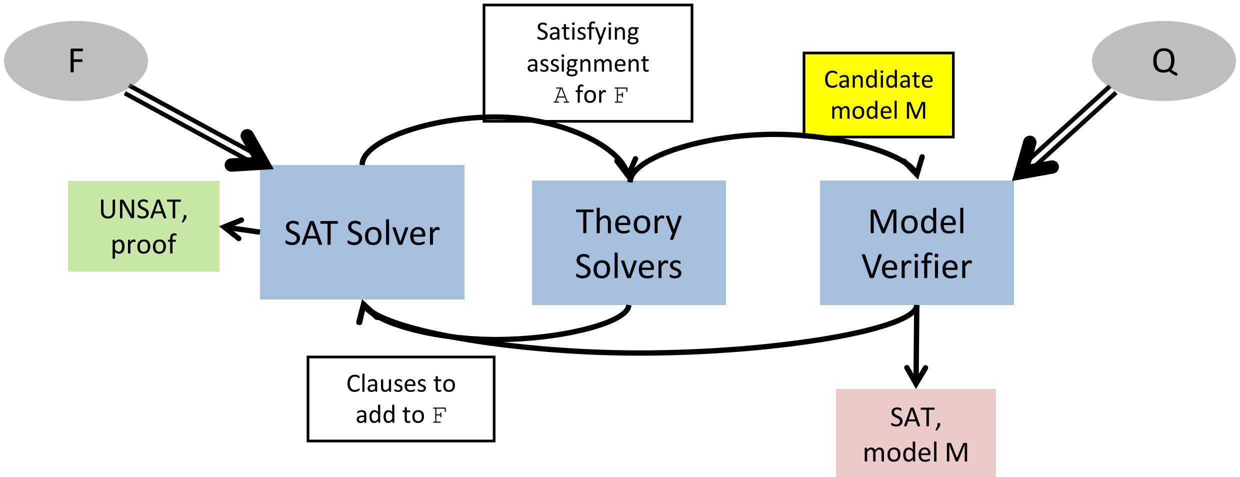
true — ¬salesman(person$_2$)

true — salesman(person$_3$)

$\forall$ x y . salesman(x) $\Rightarrow$ travels(x,y)

- *A is Theory-Consistent according to the theory of equality*

Construct Candidate Model $\mathbb{M}$ from $\mathbb{A}$



$\mathbb{A}$:=

{ distinct(NewYork, Boston, Seattle),

travels(person$_1$, Boston),

$\neg$salesman(person$_2$),

salesman(person$_3$) }

$\Rightarrow$

$\mathbb{M}$:=

travels :

person$_1$, Boston $\rightarrow$ true

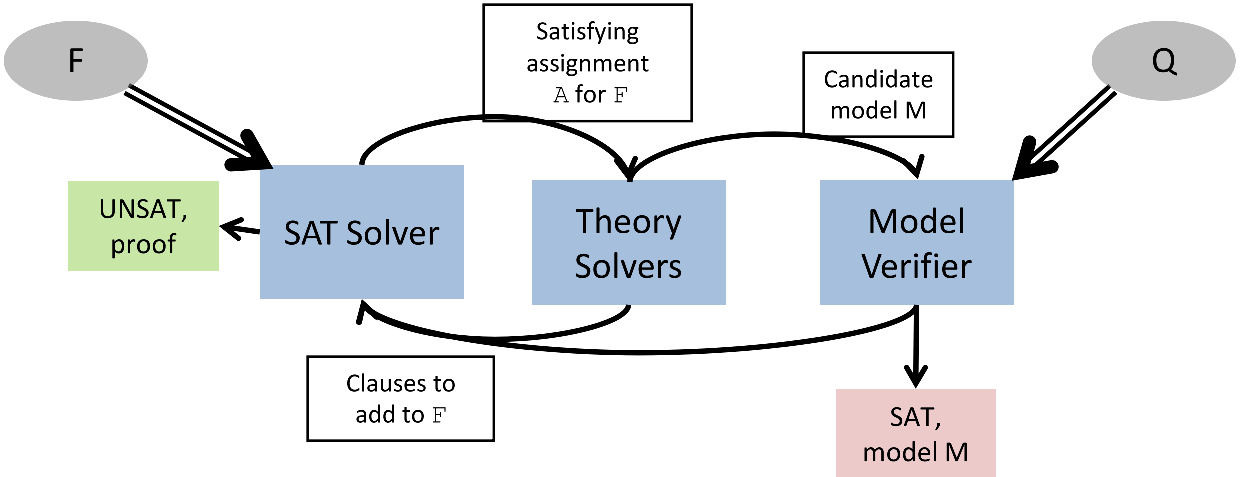
... $\rightarrow$ false

salesman :

person$_3$ $\rightarrow$ true

... $\rightarrow$ false

Determine if M satisfies Q



M :=

travels :

person$_1$, Boston $\rightarrow$ true

... $\rightarrow$ false

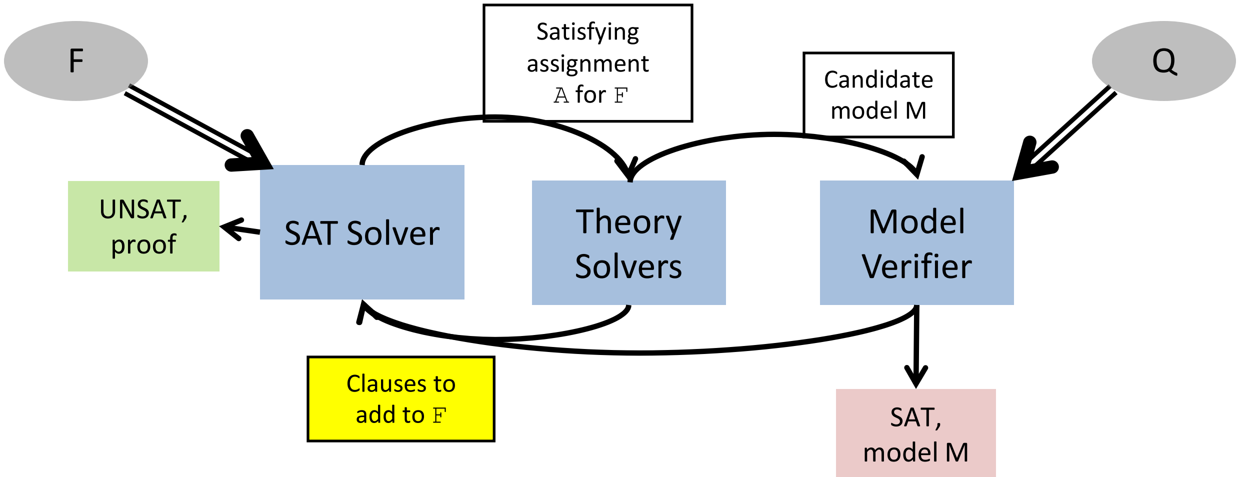
salesman :

person$_3$ $\rightarrow$ true

... $\rightarrow$ false

Q : $\forall$xy. salesman(x) $\Rightarrow$ travels(x,y)

Add Clauses back to $\mathbb{F}$



$\mathbb{M}$:=

travels :

person$_1$, Boston $\rightarrow$ true

… $\rightarrow$ false

salesman :

person$_3$ $\rightarrow$ true

… $\rightarrow$ false

$\mathbb{Q}$: $\forall$xy. salesman(x) $\Rightarrow$ travels(x,y)

Ψ[x, y]

- *Ψ is false for person$_3$, NewYork*
- Add Ψ[person$_3$, NewYork] to $\mathbb{F}$
- Will rule out $\mathbb{M}$ on next iteration
  - Model "refinement" process

Finding Small Models : Motivation

$\mathbb{M}$:=

Person : { $person_1$, $person_2$, $person_3$ }

City : { NewYork, Boston, Seattle }

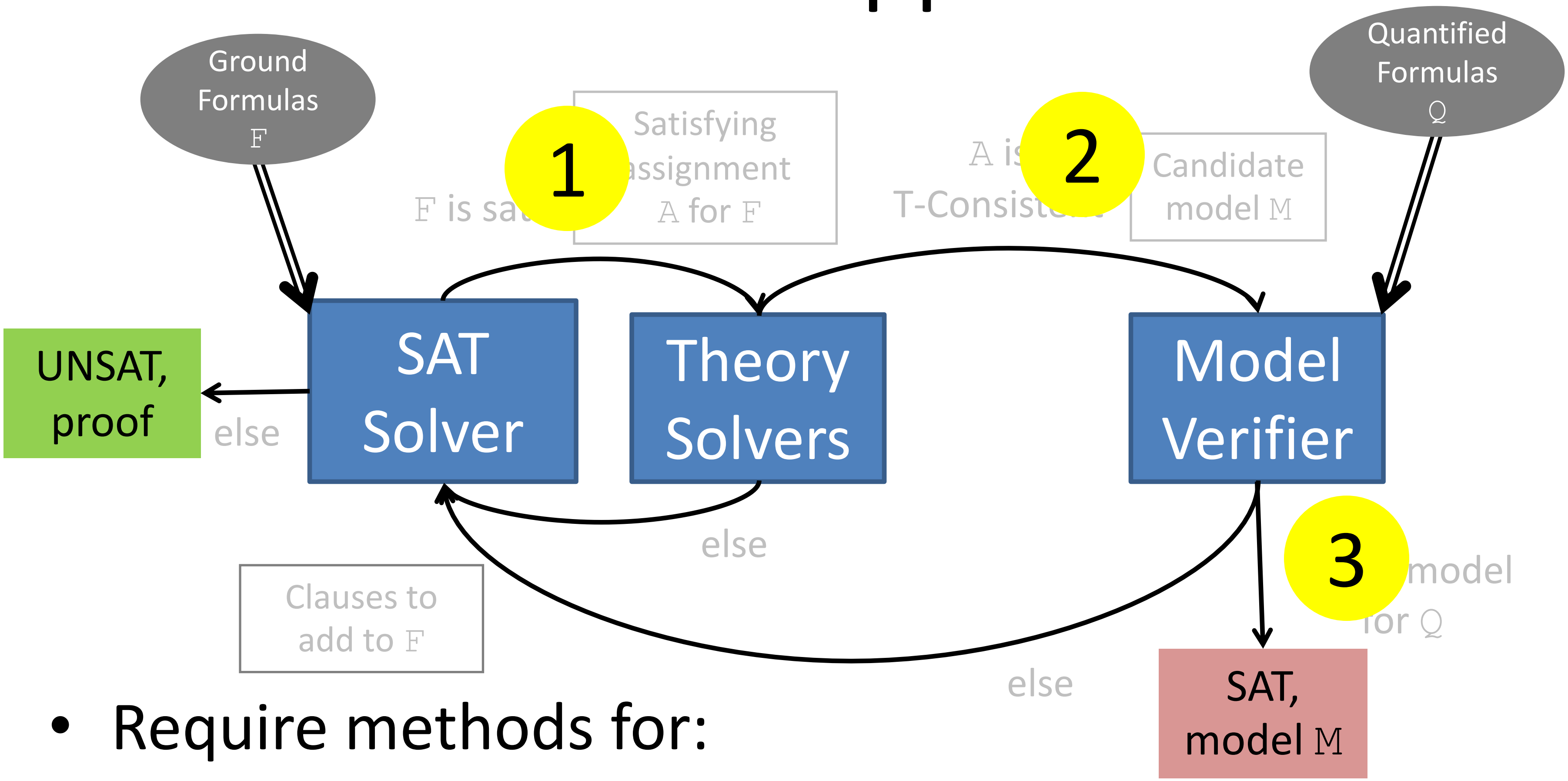
…

$\mathbb{Q}$: $\forall$x : Person, y : City.

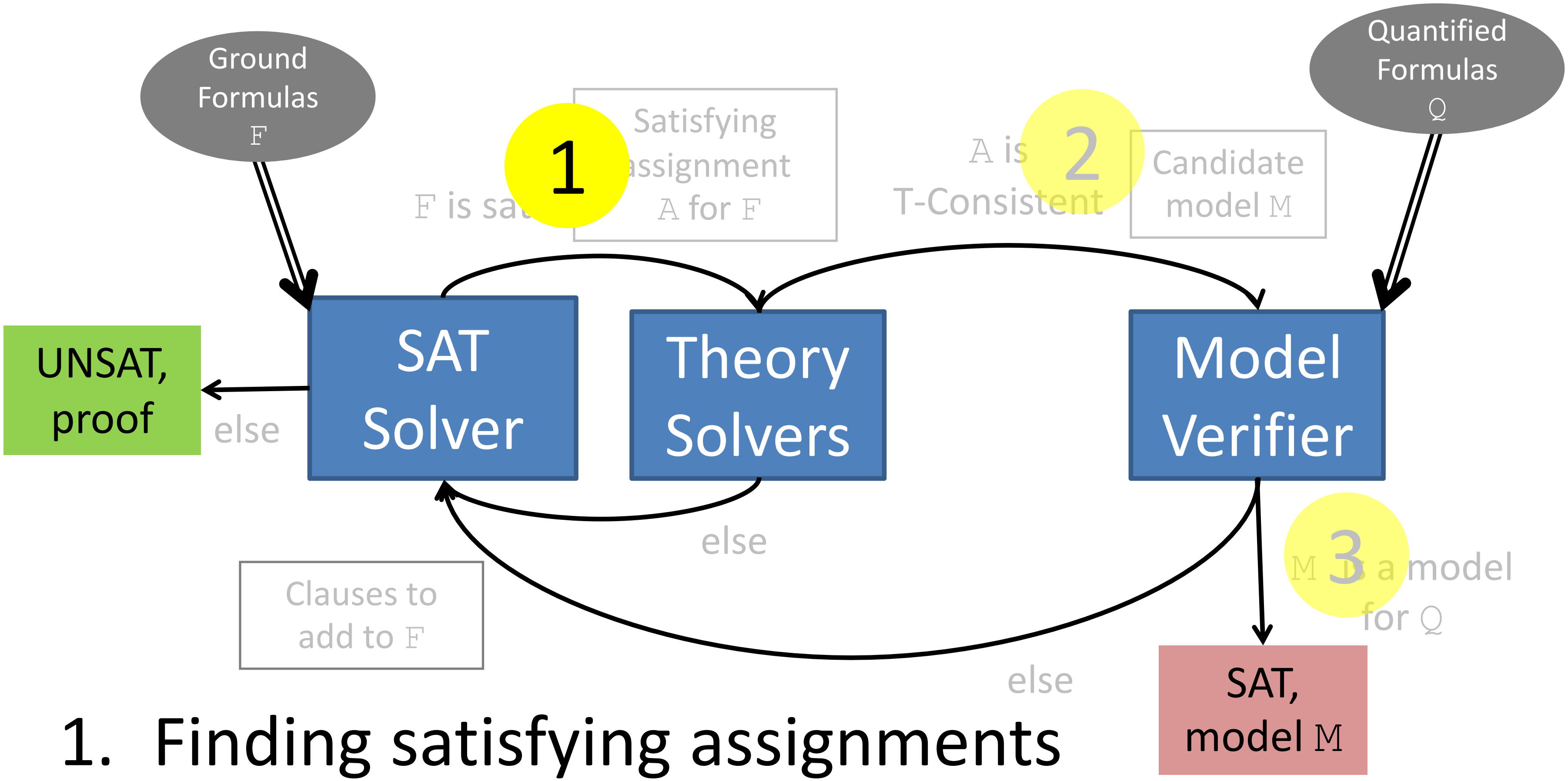
salesman(x) $\Rightarrow$ travels(x,y)

- Naïvely, to determine whether $\mathbb{M}$ is model for $\mathbb{Q}$:
  - Check if $\mathbb{M}$ satisfies *all instances* $\mathbb{S}$ of $\mathbb{Q}$
- Challenge: $\mathbb{S}$ can be very large
  - For $\mathbb{Q}$ with n vars, domain size d, $|\mathbb{S}|$ can be O(d^n)
    - In example, 3 * 3 = 9
- Solution:
  - Search for candidate models with small domain sizes
    - Use finite cardinality constraints [CAV 2013]

Outline of Approach

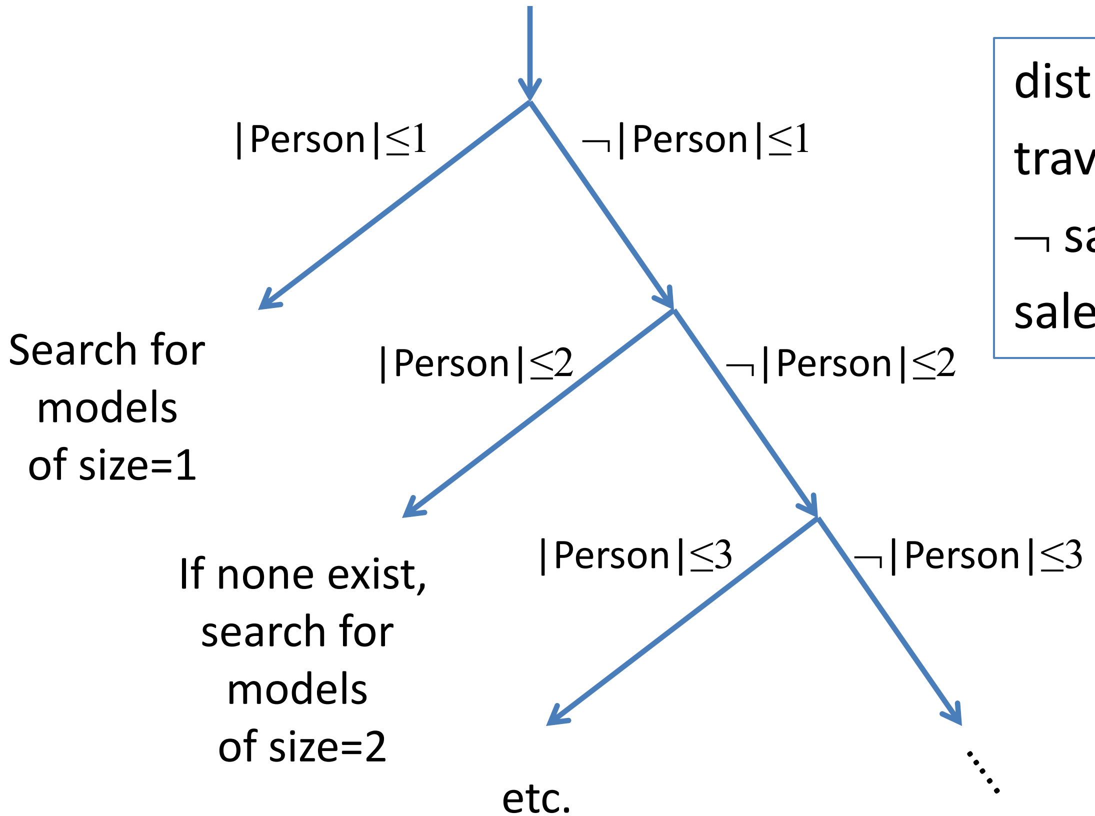


- Require methods for:
  1. Finding satisfying assignments
     - Esp. ones that induce models with small domain sizes
  2. Building candidate models
  3. Checking candidate models



1. Finding satisfying assignments
2. Building candidate models
3. Checking candidate models

Finding Minimal Models in DPLL(T)

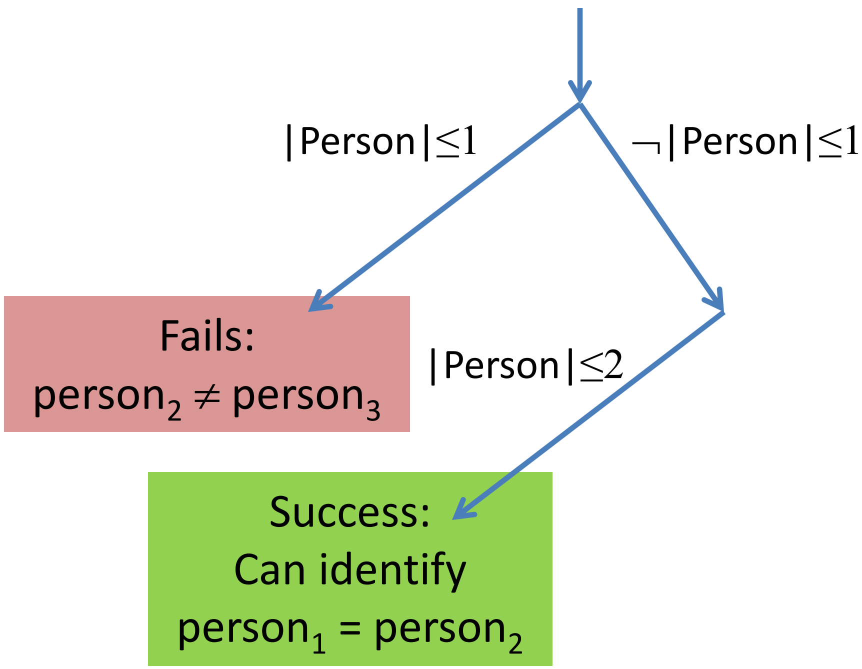


distinct(NewYork, Boston, Seattle)
travels($person_1$, Boston)
¬ salesman($person_2$)
salesman($person_3$)

- Idea: fix domain sizes incrementally 1,2,3,….

  ⇒ *Fixed-Cardinality DPLL(T)*

Finding Minimal Models in DPLL(T)



distinct(NewYork, Boston, Seattle)

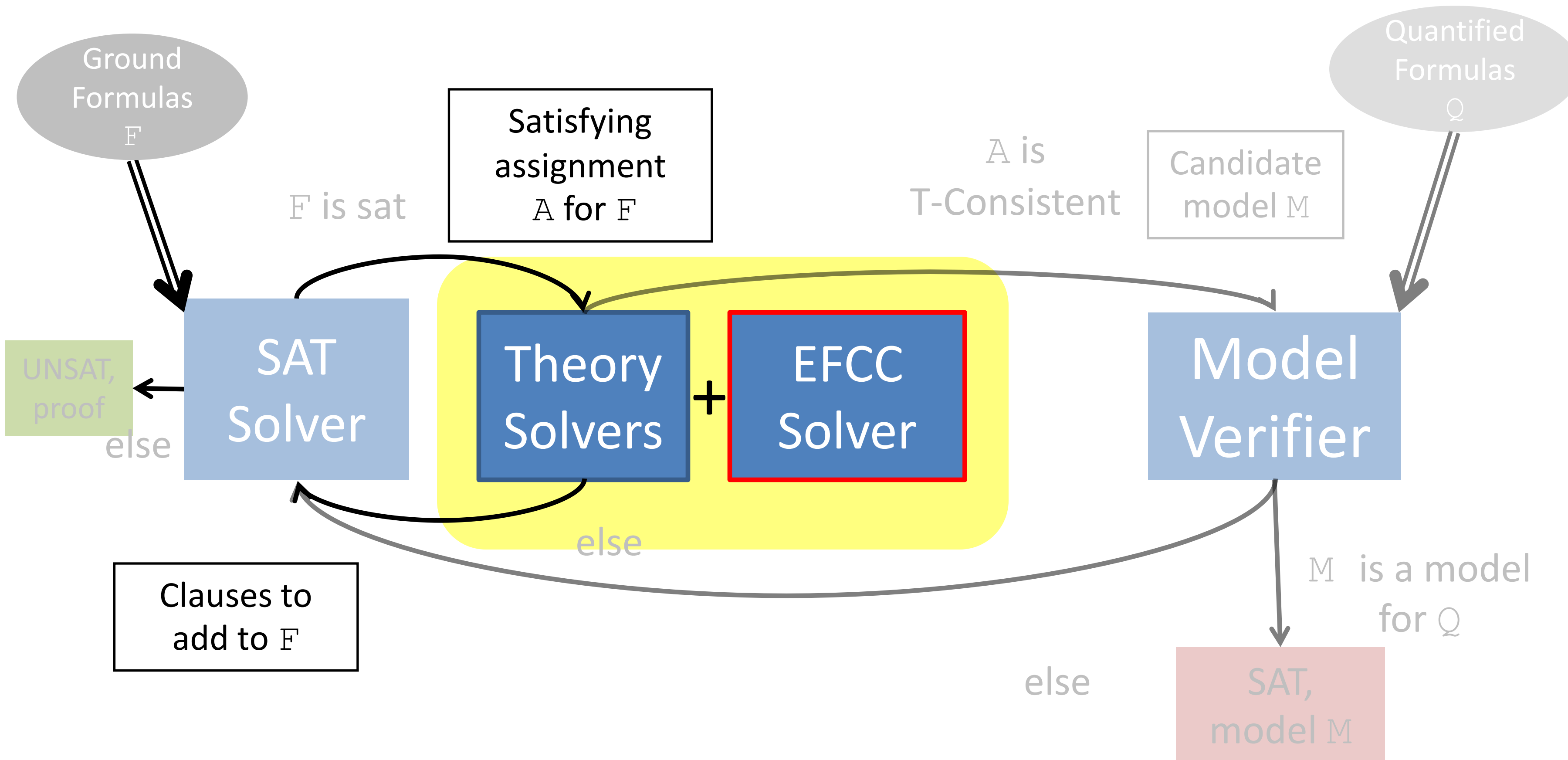
travels($person_1$, Boston)

¬ salesman($person_2$)

salesman($person_3$)

- Requires: method to find cardinality conflicts
  - E.g. determine when > 1,2,3,… “Person” must exist

Finding Minimal Models in DPLL(T)

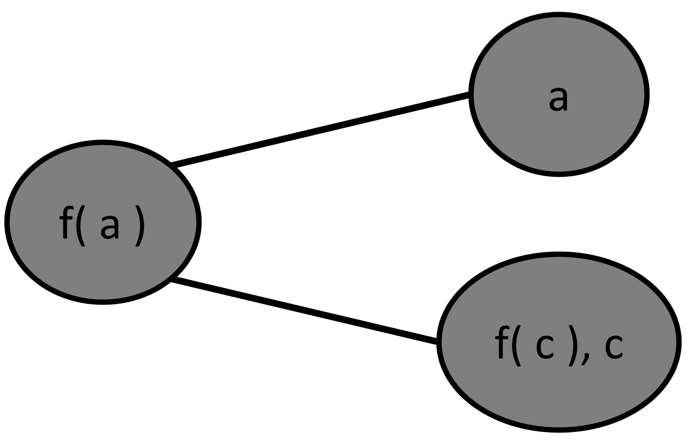


- Extend SMT solver with theory solver for:

  ⇒ *Theory of EUF + finite cardinality constraints (EFCC)*

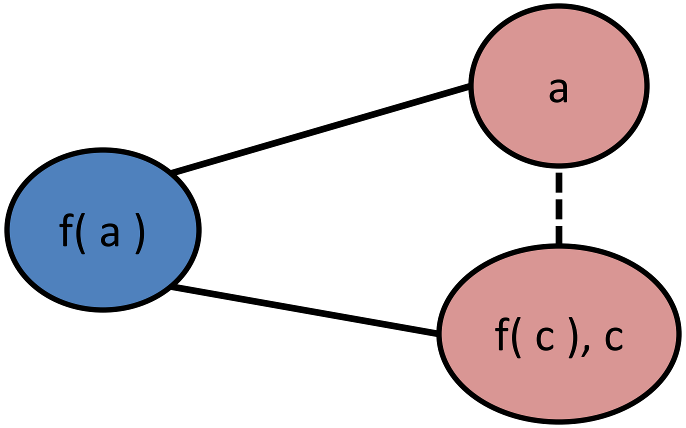
Theory Solver for EFCC

- Interested in models M where:
  - Domain elements of M are equivalence classes of terms
- Thus, signature of EFCC has predicates of form $|S| \leq k$
  - Satisfied iff ≤ k equivalence classes of terms of sort S exist
- To check if cardinality constraints are satisfied:
  - Based on *disequality graph* (V, E)
    - Vertices V are equivalence classes of sort S
    - Edges E are disequalities between terms of sort S
  - So, f(a) ≠ a, f(a) ≠ c, f(c) = c becomes:



Theory of EFCC and k-Colorability

- Assume a single sort S with cardinality k
  - Check if corresponding (V,E) is k-colorable
    - If no, then report a cardinality conflict ($C \Rightarrow \neg |S| \leq k$)
      - where C is an *explanation* of subgraph that is not k-colorable
    - If yes, we *cannot* be sure that a model of size k exists
      - Due to theory reasoning:



$|S| \leq 2$

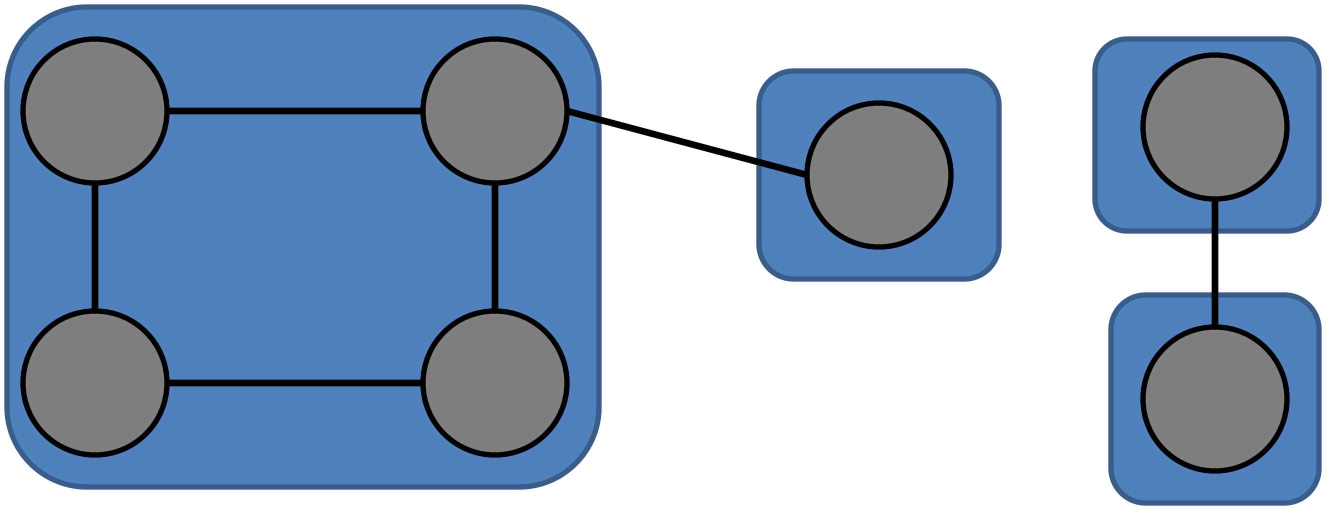
$\Rightarrow$ *Must explicitly merge equivalence classes*

Theory of EFCC : Challenges

- Why finite cardinality constraints are challenging:
  - Interaction with theory reasoning
  - k-colorability is NP-complete
  - Analysis must be incremental
- Solution:
  - Explicitly merge equivalence classes
  - Use heuristic region-based approach which:
    - May quickly detect when disequality graph is not k-colorable
    - Suggests pairs of equivalence classes to merge

Region-Based Approach

- Partition the graph (V, E) into *regions* with high edge density



$|S| \leq 2$

- For $|S| \leq k$ we maintain the invariant:
  - No clique of size $k+1$ exists having nodes from multiple regions
- Thus, we only need to search for cliques local to regions
  - Region can be ignored if it has ≤ k nodes

Extension to Multiple Sorts

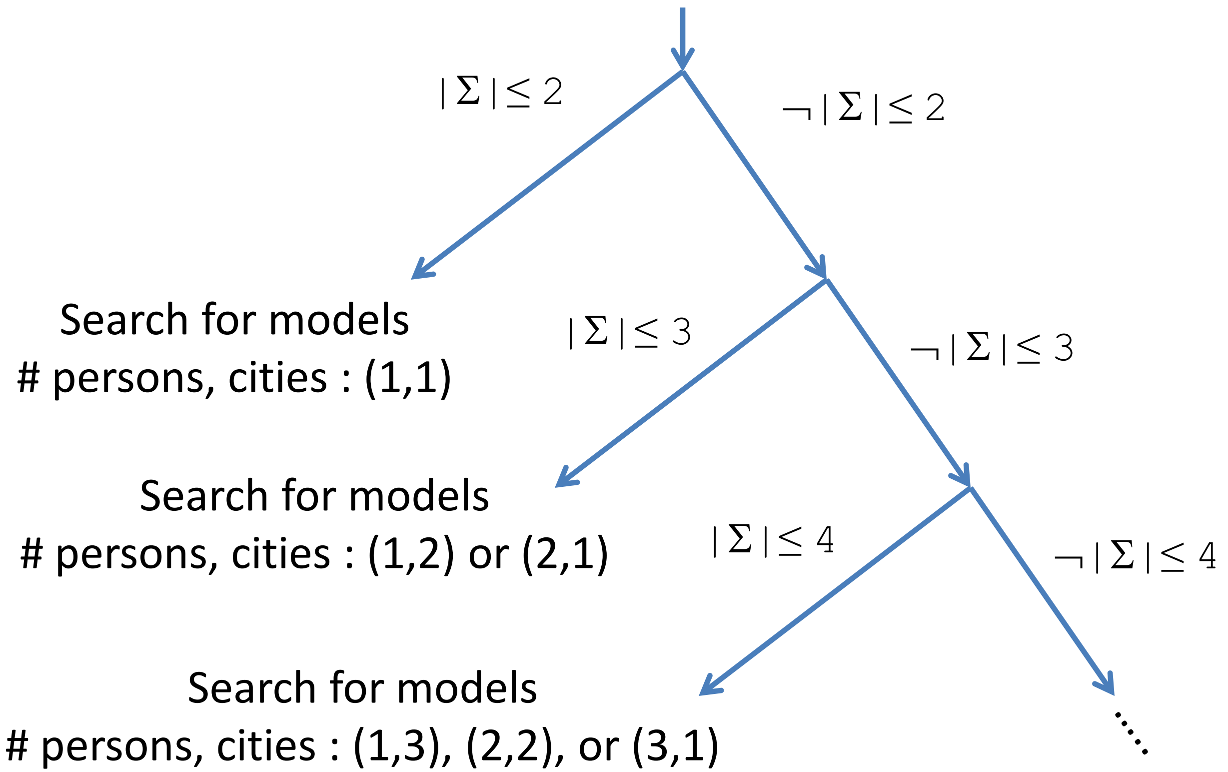
- Challenge:
  - Fair Strategy for enumerating cardinalities
- Example:

$person_1 \neq person_2 \vee (city_1 \neq city_2 \ldots city_1 \neq city_{1000} \ldots city_{999} \neq city_{1000})$

  - Formula has model with 2 persons, 1 city
  - But we may search for models where
    - # persons, cities : (1, 1), (1, 2), …., (1,1000)
- With quantified formulas, this leads to *incompleteness*
  - May imply no finite models exist in a branch

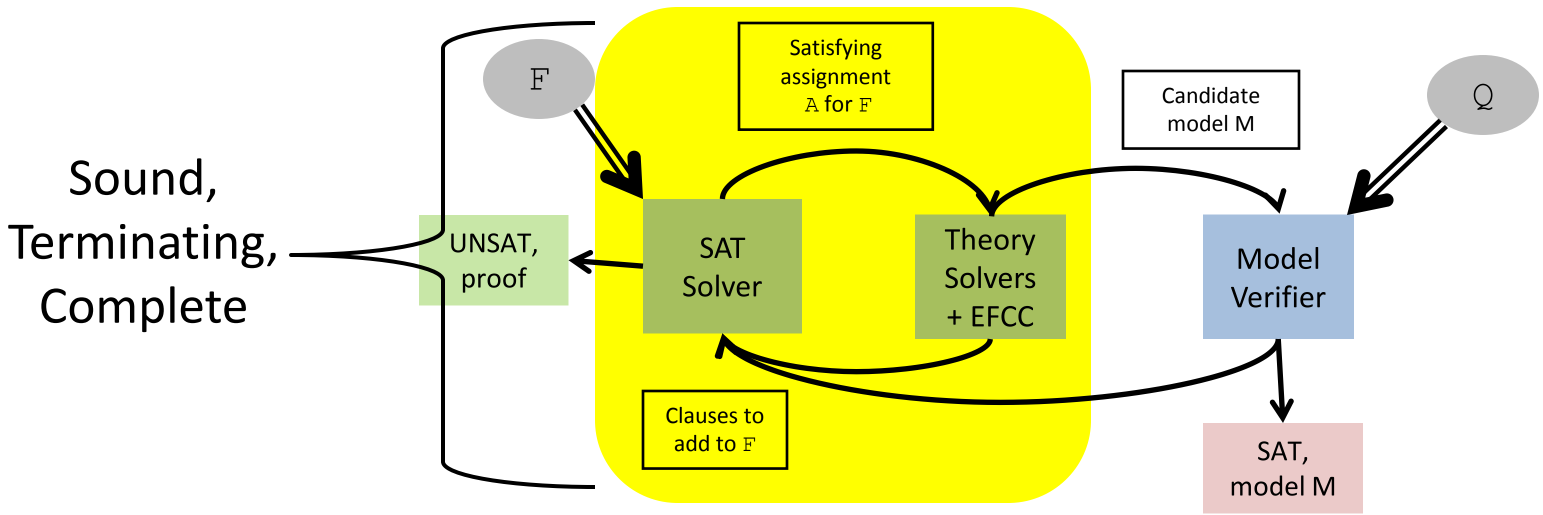
Fixed-Cardinality DPLL(T) for Multiple Sorts

- Uses extended signature containing:
  - Boolean predicates of form $|\Sigma| \leq k$
    - Satisfied if and only if ≤ k equivalence classes *for all* sorts exist

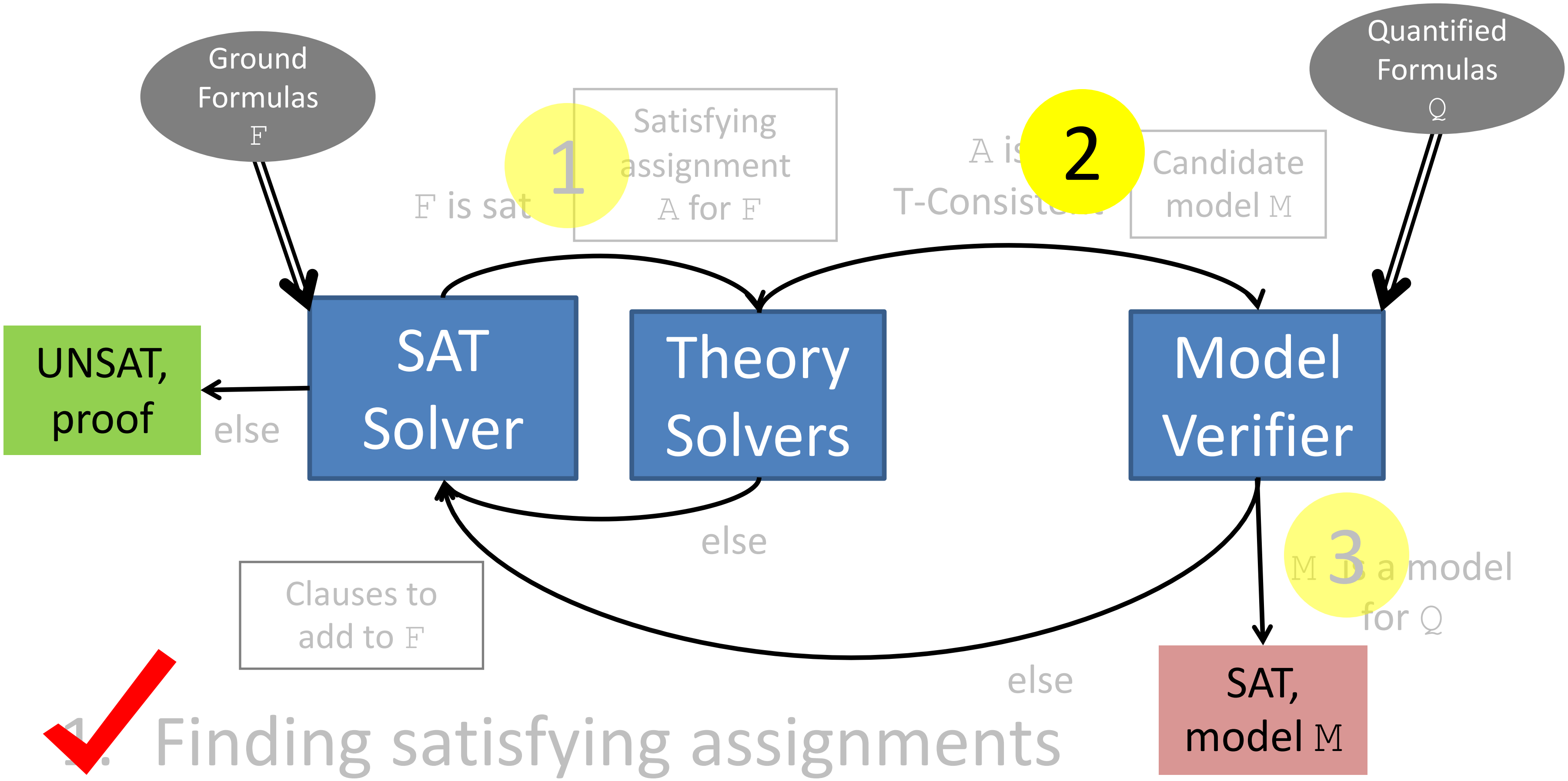


$\Rightarrow$ Gives a fair strategy

Properties : Ground Solver



- For ground inputs F,
  - Fixed-cardinality DPLL(T), using Theory EFCC:
    - Sound, terminating, and complete
      - Eventually either:
        - » Determines F is unsatisfiable
        - » Constructs candidate model M of finite (minimal) size



1. Finding satisfying assignments
2. Building candidate models
3. Checking candidate models

Model Representation

- Represent a function/predicate as a list of entries:

$$C_1 \rightarrow v_1, \ldots, C_n \rightarrow v_n$$

  - Where
    - $C_1, \ldots, C_n$ are "conditions"
    - $v_1, \ldots, v_n$ are "values"
- E.g. unary predicate "P" true only for v represented as:

$$(\ v\ ) \rightarrow T,\ (\ *\ ) \rightarrow \bot$$

  - Interpreted as an if-then-else:

$$\lambda\ x.\ ite(\ x = v, T, \bot\ )$$

Model Construction

- Candidate models $\mathbb{M}$:
  - Domain elements are equivalence classes $[t_1]$, $[t_2]$, …
  - Are constructed from sat assignment $\mathbb{A}$ for $\mathbb{F}$
  - Consist of definitions D_f for each $f \in \Sigma$, where each D_f:
    - Is partially determined by ground equalities from $\mathbb{A}$
      - For each equality $f(t_1, …, t_n) = t$ in $\mathbb{A}$,
        - » Entry $([t_1], …, [t_n]) \rightarrow [t] \in D_f$
    - Has default value
      - Determined by *distinguished f-application e*
        - » Entry $(*, …, *) \rightarrow [e] \in D_f$

Constructing Models : Example

$person_1$, $person_2$, $person_3$: Person
NewYork, Boston, Seattle : City
salesman: Person $\rightarrow$ Bool
travels : Person $\times$ City $\rightarrow$ Bool

$\mathbb{F}$

distinct(NewYork, Boston, Seattle)
$\neg$travels($person_1$, Boston)
$\neg$salesman($person_2$)
salesman($person_3$)
salesman($person_1$) $\Rightarrow$ travels($person_1$, NewYork)

$\mathbb{Q}$

$\forall$ x y . salesman(x) $\Rightarrow$ travels(x,y)

- Guide choice of default values based on :
  - $person_1$ for Person
  - NewYork for City
- Assume $\mathbb{Q}$ has been instantiated with these terms

Constructing Models : Example

$person_1, person_2, person_3$: Person
NewYork, Boston, Seattle : City
salesman: Person $\rightarrow$ Bool
travels : Person $\times$ City $\rightarrow$ Bool

$\mathbb{F}$

- distinct(NewYork, Boston, Seattle) } true
- travels($person_1$, Boston) } true
- $\neg$salesman($person_2$) } true
- salesman($person_3$) } true

salesman($person_1$) $\Rightarrow$ travels($person_1$, NewYork) } true

$\mathbb{Q}$

$\forall$ x y . salesman(x) $\Rightarrow$ travels(x,y)

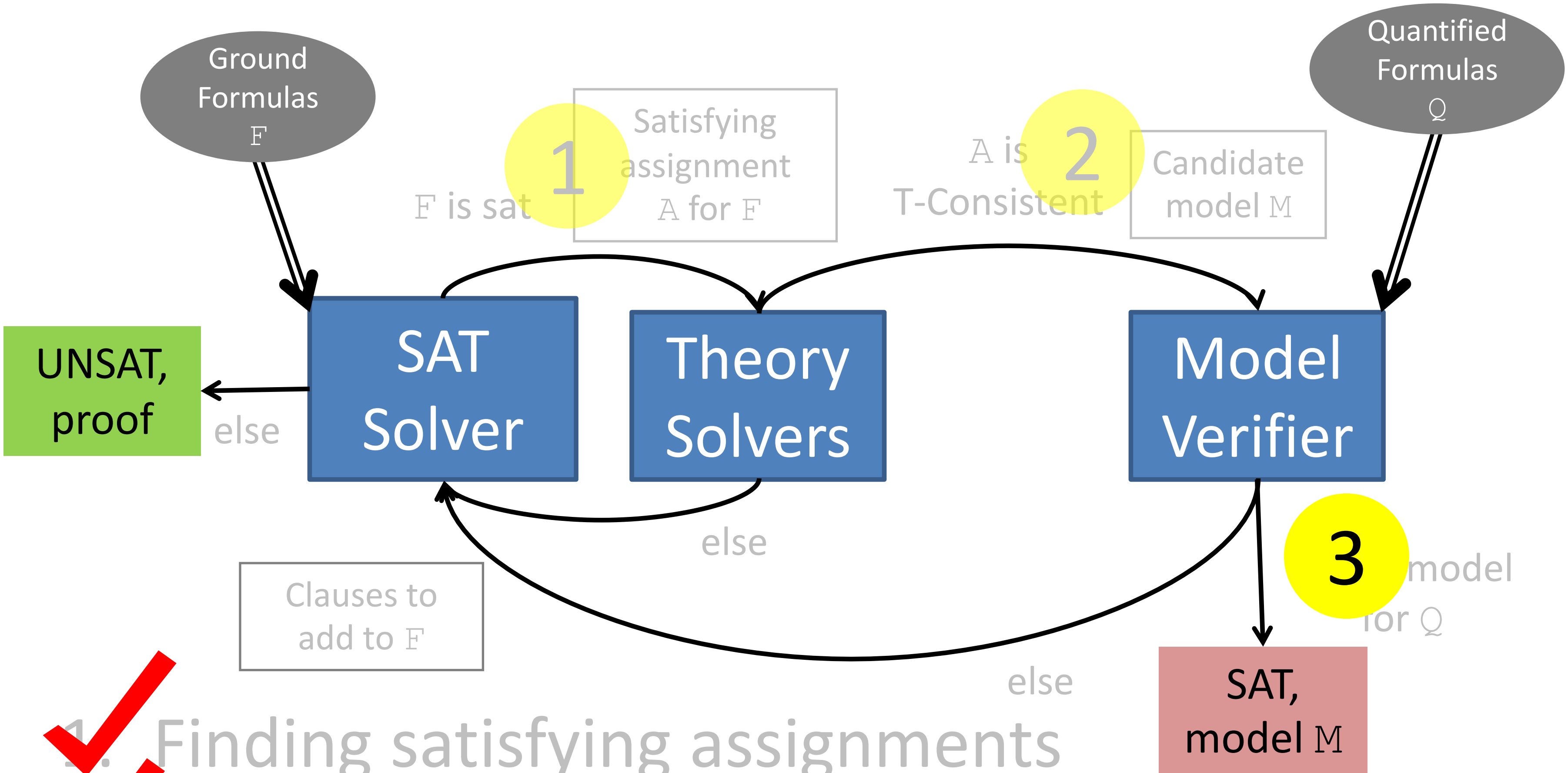
- Choose default based on value of travels($person_1$, NewYork)

$\mathbb{A}$:=

{ …,
travels($person_1$, Boston) = T,
travels($person_1$, NewYork) = T }

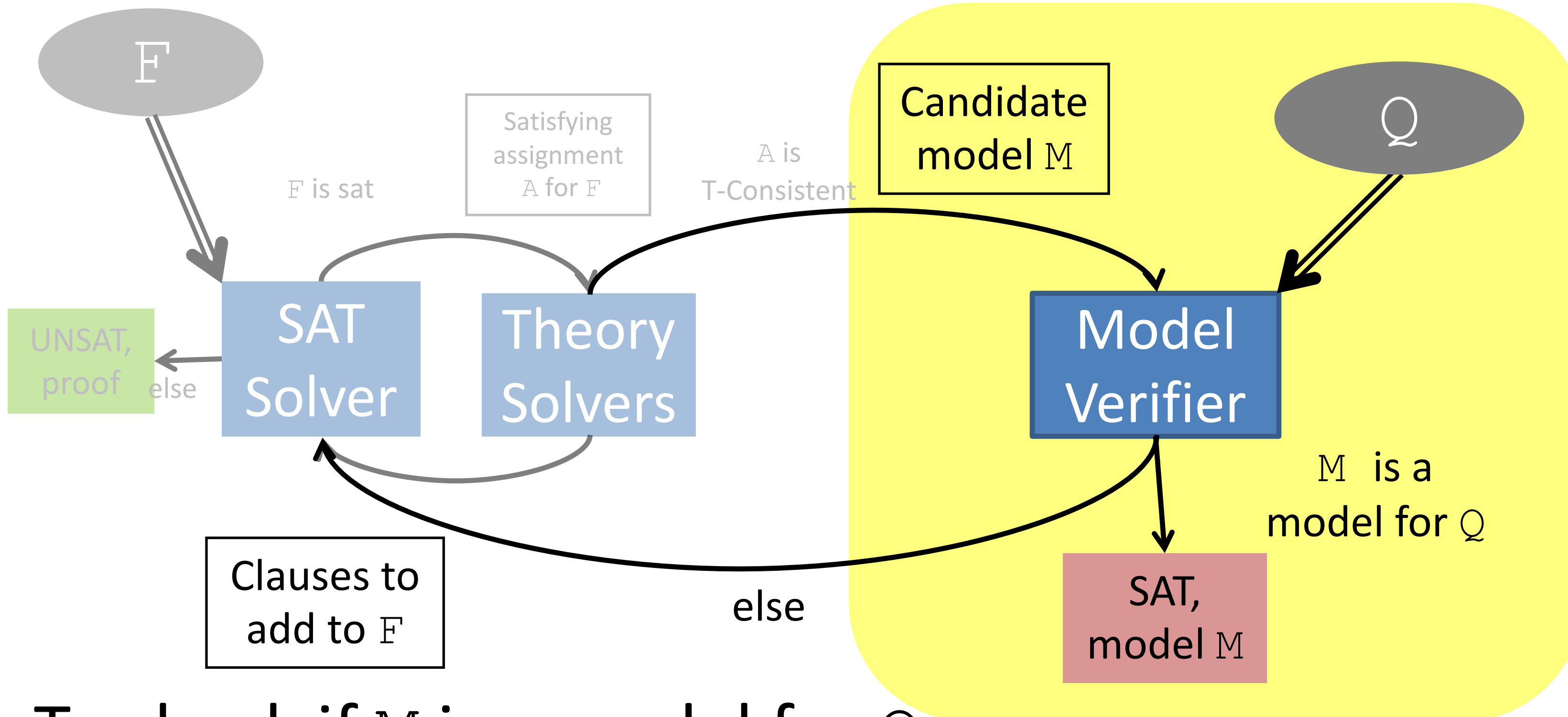
$D_{travels}$:

($person_1$, NewYork) $\rightarrow$ T,
($person_1$, Boston) $\rightarrow$ $\bot$,
(*, *) $\rightarrow$ T



1. Finding satisfying assignments
2. Building candidate models
3. Checking candidate models

Checking Candidate Models



- To check if $\mathtt{M}$ is a model for $\mathtt{Q}$:
  - Naïvely, test if every instance of $\mathtt{Q}$ is true in $\mathtt{M}$
  - Or, choose a *representative* set of instances of $\mathtt{Q}$
    - Only add instances that are false in $\mathtt{M}$
    - Identify sets of instances of $\mathtt{Q}$ that are equisatisfiable

Checking Candidate Models

$person_1, person_2, person_3$: Person
NewYork, Boston, Seattle : City
salesman: Person $\rightarrow$ Bool
travels : Person $\times$ City $\rightarrow$ Bool

$\mathbb{F}$

distinct(NewYork, Boston, Seattle)
$\neg$travels($person_1$, Boston)
$\neg$salesman($person_2$)
salesman($person_3$)
salesman($person_1$) $\Rightarrow$ travels($person_1$, NewYork)

$\mathbb{Q}$

$\forall$ x y . salesman(x) $\Rightarrow$ travels(x,y)

$D_{salesman}$:

($person_2$) $\rightarrow \bot$,
($person_3$) $\rightarrow$ T,
(*) $\rightarrow$ T

$D_{travels}$:

($person_1$, NewYork) $\rightarrow$ T
($person_1$, Boston) $\rightarrow \bot$,
(*, *) $\rightarrow$ T }

$\mathbb{Q}$[$person_1$, NewYork]
$\mathbb{Q}$[$person_1$, Boston]
$\mathbb{Q}$[$person_1$, Seattle]
$\mathbb{Q}$[$person_2$, NewYork]
$\mathbb{Q}$[$person_2$, Boston]
$\mathbb{Q}$[$person_2$, Seattle]
$\mathbb{Q}$[$person_3$, NewYork]
$\mathbb{Q}$[$person_3$, Boston]
$\mathbb{Q}$[$person_3$, Seattle]

Checking Candidate Models

$person_1, person_2, person_3 : Person$
NewYork, Boston, Seattle : City
salesman: Person $\rightarrow$ Bool
travels : Person $\times$ City $\rightarrow$ Bool

F

distinct(NewYork, Boston, Seattle)
$\neg travels(person_1, Boston)$
$\neg salesman(person_2)$
$salesman(person_3)$

$salesman(person_1) \Rightarrow travels(person_1, NewYork)$

Q

$\forall\ x\ y\ .\ salesman(x) \Rightarrow travels(x,y)$

$D_{salesman}$:

$(person_2) \rightarrow \bot$,
$(person_3) \rightarrow T$,
$(*) \rightarrow T$

$D_{travels}$:

$(person_1, NewYork) \rightarrow T$
$(person_1, Boston) \rightarrow \bot$,
$(*, *) \rightarrow T$ }

$Q[person_1, NewYork]$ — true
$Q[person_1, Boston]$ — false
$Q[person_1, Seattle]$ — true
$Q[person_2, NewYork]$
$Q[person_2, Boston]$
$Q[person_2, Seattle]$ — true
$Q[person_3, NewYork]$
$Q[person_3, Boston]$
$Q[person_3, Seattle]$ — true

Checking Candidate Model $\mathtt{M}$

- To check if $\mathtt{M}$ satisfies quantified formula $\mathtt{Q}$:
  - Choose representative set of instances $\mathtt{S}$ of $\mathtt{Q}$
    - $\Rightarrow$ This is somewhat heuristic
  - For each Ψ in $\mathtt{S}$,
    - If $\mathtt{M}(\Psi) = \mathtt{false}$, add Ψ to $\mathtt{F}$
  - If no instances added, then $\mathtt{M}$ satisfies $\mathtt{Q}$

- Alternate, improved approach :
  - Directly compute the interpretation of $\mathtt{Q}$ in $\mathtt{M}$
    - Using same data structure that represents functions in $\mathtt{M}$

Computing Interpretations of Terms

$\mathbb{Q} : \forall\, x\, y\, .\, \text{salesman}(x) \Rightarrow \text{travels}(x,y)$

$D_{\text{salesman}(x)}$:

$(\text{person}_2, *) \rightarrow \bot,$

$(\text{person}_3, *) \rightarrow \text{T},$

$(*, *) \rightarrow \text{T}$

$D_{\text{travels}(x,y)}$:

$(\text{person}_1, \text{NewYork}) \rightarrow \text{T}$

$(\text{person}_1, \text{Boston}) \rightarrow \bot,$

$(*, *) \rightarrow \text{T} \quad \}$

Computing Interpretations of Terms

$Q : \forall\ x\ y\ .\ salesman(x) \Rightarrow travels(x,y)$

$D_{salesman(x)}$:

$($person$_2$, *$) \rightarrow \bot$,
$($person$_3$, *$) \rightarrow$ T,
$(*, *) \rightarrow$ T

X

$D_{travels(x,y)}$:

$($person$_1$, NewYork$) \rightarrow$ T
$($person$_1$, Boston$) \rightarrow \bot$,
$(*, *) \rightarrow$ T }

=

$D_{salesman(x)} \times D_{travels(x,y)}$:

$($person$_2$, *$) \rightarrow (\bot$, T$)$,
$($person$_3$, *$) \rightarrow ($T, T$)$,
$($person$_1$, NewYork$) \rightarrow ($T, T$)$
$($person$_1$, Boston$) \rightarrow ($T, $\bot)$,
$(*, *) \rightarrow ($T, T$)$

Compute product

Computing Interpretations of Terms

$\mathcal{Q} : \forall\ x\ y\ .\ \text{salesman}(x) \Rightarrow \text{travels}(x,y)$

$$\left(\begin{array}{l} D_{salesman(x)}: \\ (person_2, *) \to \bot, \\ (person_3, *) \to T, \\ (*, *) \to T \end{array} \quad \mathbf{X} \quad \begin{array}{l} D_{travels(x,y)}: \\ (person_1, NewYork) \to T \\ (person_1, Boston) \to \bot, \\ (*, *) \to T \ \} \end{array} \right)$$

$$\|$$

$$\Rightarrow \left(\begin{array}{l} D_{salesman(x)} \times D_{travels(x,y)}: \\ (person_2, *) \to (\bot, T), \\ (person_3, *) \to (T, T), \\ (person_1, NewYork) \to (T, T) \\ (person_1, Boston) \to (T, \bot), \\ (*, *) \to (T, T) \end{array} \right) = \begin{array}{l} D_{salesman(x) \Rightarrow travels(x,y)}: \\ (person_2, *) \to (\bot \Rightarrow T), \\ (person_3, *) \to (T \Rightarrow T), \\ (person_1, NewYork) \to (T \Rightarrow T), \\ (person_1, Boston) \to (T \Rightarrow \bot), \\ (*, *) \to (T \Rightarrow T) \end{array}$$

Apply interpreted predicate

Computing Interpretations of Terms

$\mathbb{Q} : \forall\ x\ y\ .\ salesman(x) \Rightarrow travels(x,y)$

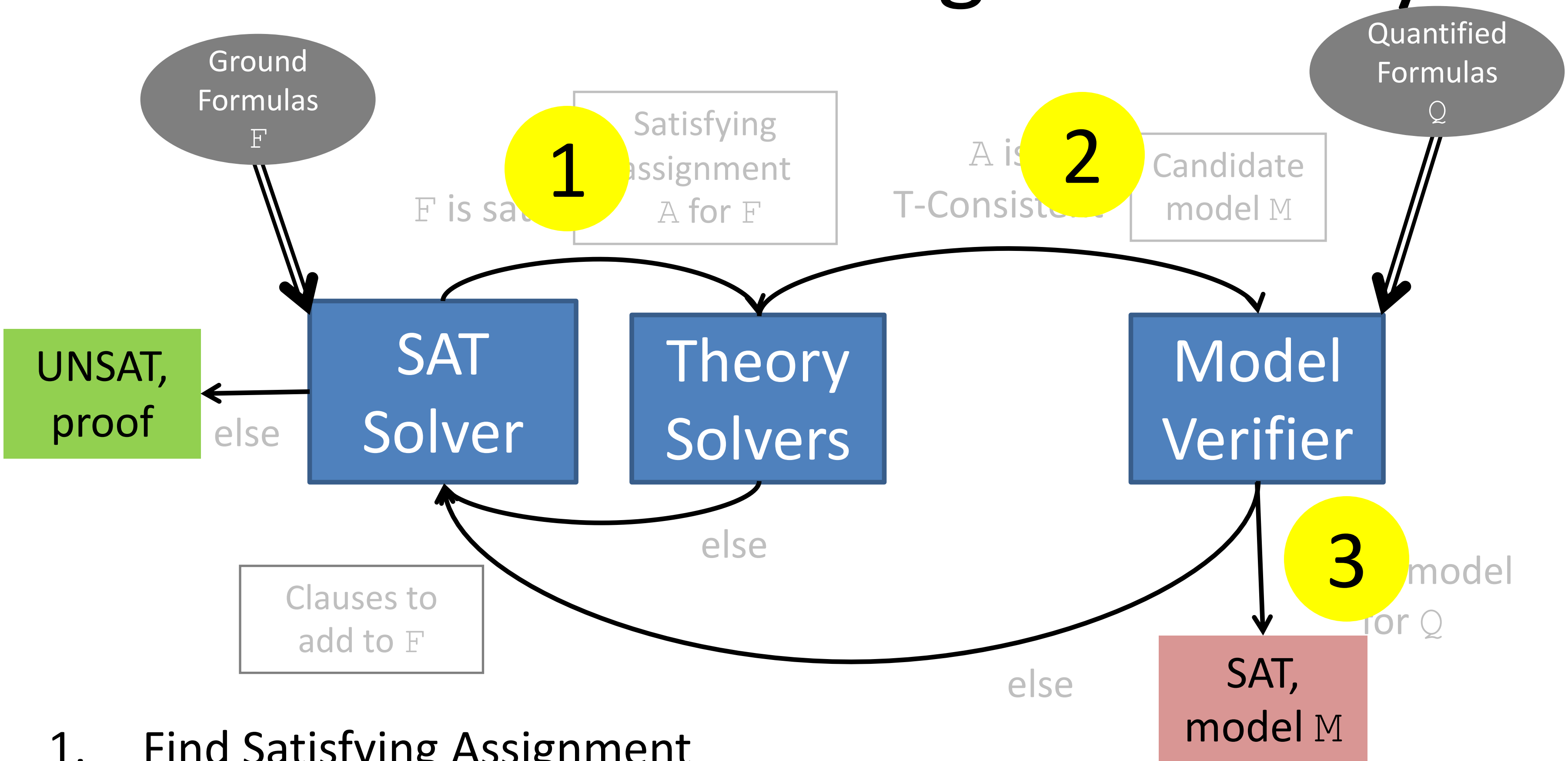
$$\left(\begin{array}{l} D_{salesman(x)}: \\ \quad (person_2, *) \to \bot, \\ \quad (person_3, *) \to T, \\ \quad (*, *) \to T \end{array} \quad \times \quad \begin{array}{l} D_{travels(x,y)}: \\ \quad (person_1, NewYork) \to T \\ \quad (person_1, Boston) \to \bot, \\ \quad (*, *) \to T \quad \} \end{array} \right)$$

$$\|$$

$$\Rightarrow \left(\begin{array}{l} D_{salesman(x)} \times D_{travels(x,y)}: \\ \quad (person_2, *) \to (\bot, T), \\ \quad (person_3, *) \to (T, T), \\ \quad (person_1, NewYork) \to (T, T) \\ \quad (person_1, Boston) \to (T, \bot), \\ \quad (*, *) \to (T, T) \end{array} \right) = \begin{array}{l} D_{salesman(x) \Rightarrow travels(x,y)}: \\ \quad (person_2, *) \to T, \\ \quad (person_3, *) \to T, \\ \quad (person_1, NewYork) \to T, \\ \quad (person_1, Boston) \to \bot, \\ \quad (*, *) \to T \end{array}$$

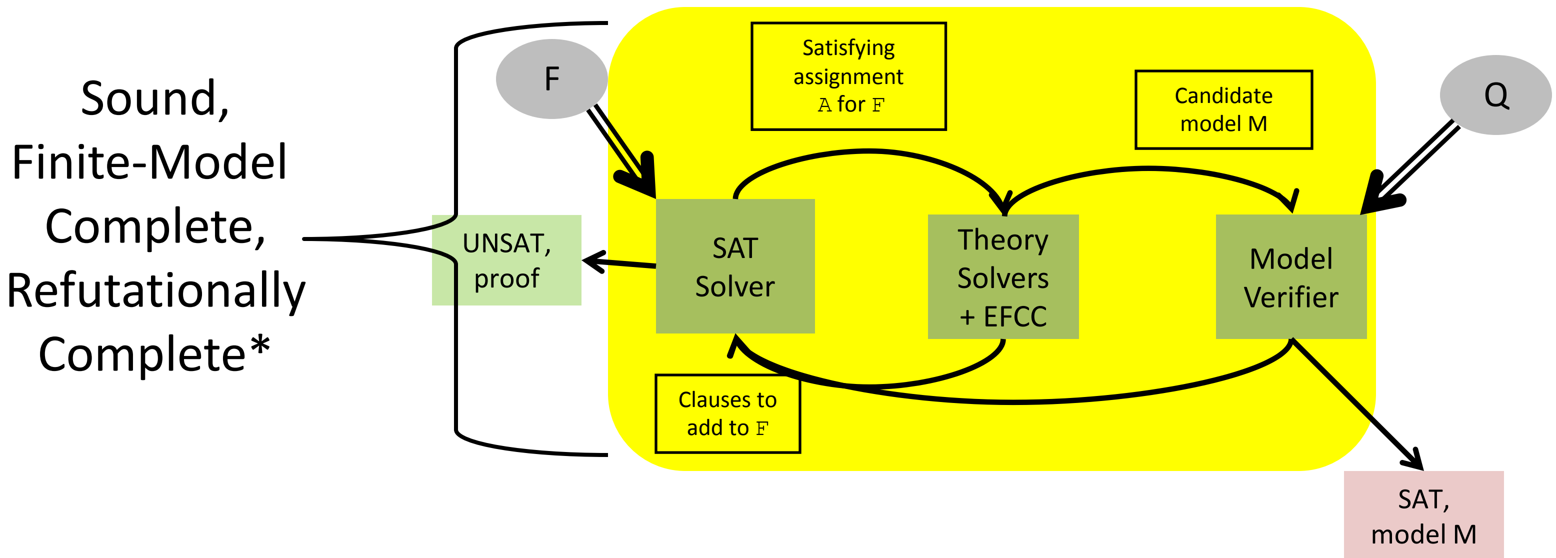
- Add $\mathbb{Q}[\ person_1/x,\ Boston/y\ ]$ to $\mathbb{F}$

Finite Model Finding: Summary



1. Find Satisfying Assignment
   – Use EFCC Solver to find Small Candidate Models
2. Construct Candidate Models
3. Model-Based Quantifier Instantiation
   – Two methods: Generalizing Evaluations, Constructing Interpretations

Properties : Finite Model Finding



- For inputs (F, Q), quantifiers in Q over free sorts
  - Fixed-cardinality DPLL(T) + quantifier instantiation:
    - Sound
    - Finite Model Complete
      - If (F, Q) has a finite model, we will eventually answer "SAT"
    - Refutationally Complete (when containing no theory symbols)
      - If (F, Q) is unsatisfiable, we will eventually answer "UNSAT"

* - under certain restrictions

Finite Model Finding: Properties

- For unsatisfiable (F, Q), quant. of Q over free sorts
  - When (F, Q) contain theory symbols
    - Approach has weaker completeness property:
      - If there exists a set I of instances of Q where:
        - » I is finite
        - » $F \wedge I$ is UNSAT
      - Then,
        - » Fixed-cardinality DPLL(T)+QI terminates, answering UNSAT
- Thus, approach is only non-terminating when:
  - (F, Q) is SAT, but only has infinite models
  - (F, Q) is UNSAT, but all finite subsets are SAT

Enhancements

- Heuristic Instantiation
  - First see if instantiations based on heuristics exist
    - If not, resort to model-based instantiation
  - May lead to:
    - Discovering easy conflicts, if they exist
    - Arriving at model faster
      - Instantiations rule out spurious models
- Sort Inference
  - Reduce symmetries in problem
- Relevancy
  - Reduce the size of satisfying assignments

Experiments

- Implemented state of the art SMT solver CVC4
- Experiments on:
  - DVF Benchmarks
    - Taken from verification tool DVF used by Intel
    - Both SAT/UNSAT benchmarks
      - SAT benchmarks generated by removing necessary pf assumptions
    - Many theories: UF, arithmetic, arrays, datatypes
    - Quantifiers only over free sorts
      - Memory addresses, Values, Sets, …
  - TPTP Benchmarks
    - Automated theorem proving community
    - No theory reasoning
  - Isabelle Benchmarks
    - Provable and unprovable goals, contains some arithmetic

Results: DVF

SAT	german	refcount	agree	apg	bmk	Total	Time
#	45	6	42	19	37	149	
z3	45	1	0	0	0	46	8.1
cvc4+i	2	0	0	0	0	2	0.0
cvc4+f	**45**	**6**	42	18	36	147	1413.1
cvc4+fi	45	**6**	**42**	19	36	148	1333.9
cvc4+fm	**45**	**6**	42	**19**	37	149	605.4
cvc4+fmi	**45**	**6**	42	19	**37**	**149**	409.8

UNSAT	german	refcount	agree	apg	bmk	Total	Time
#	145	40	488	304	244	1221	
z3	145	**40**	**488**	304	244	**1221**	31.0
cvc4+i	**145**	40	484	**304**	**244**	1217	21.3
cvc4+f	145	40	476	298	242	1201	7512.2
cvc4+fi	145	40	488	302	244	1219	1181.4
cvc4+fm	145	40	471	300	242	1198	6949.7
cvc4+fmi	145	40	488	302	244	1219	1185.0

cvc4 :

- f : finite model
- i : heuristic
- m : model-based

600 second timeout

- CVC4 with finite model finding (cvc4+f)
  - Effective for answering SAT
  - Using heuristic instantiation, solves 4 UNSAT that cvc4 cannot

Results: TPTP

	SAT					UNSAT				
	EPR (392)	NEQ (639)	SEQ (340)	PEQ (624)	Total (1995)	EPR (1114)	NEQ (1594)	SEQ (7875)	PEQ (2003)	Total (12586)
z3	320	155	164	249	888	**989**	**412**	**3310**	1320	**6031**
cvc3	27	0	0	0	27	787	381	3019	883	5070
iprover	**363**	128	107	396	994	835	105	2690	**1523**	5153
iprover+f	362	226	178	468	1234	213	1	121	48	383
paradox	340	**304**	185	**526**	**1355**	723	17	339	186	1265
cvc4+i	32	0	0	0	32	821	383	3152	1045	5401
cvc4+f	295	178	143	375	991	759	247	887	651	2544
cvc4+fm	298	221	178	391	1088	759	169	1010	703	2641
cvc4+fM	301	235	**200**	395	1131	759	198	1073	733	2763
cvc4+fMi	292	207	153	385	1037	762	236	1281	746	3025

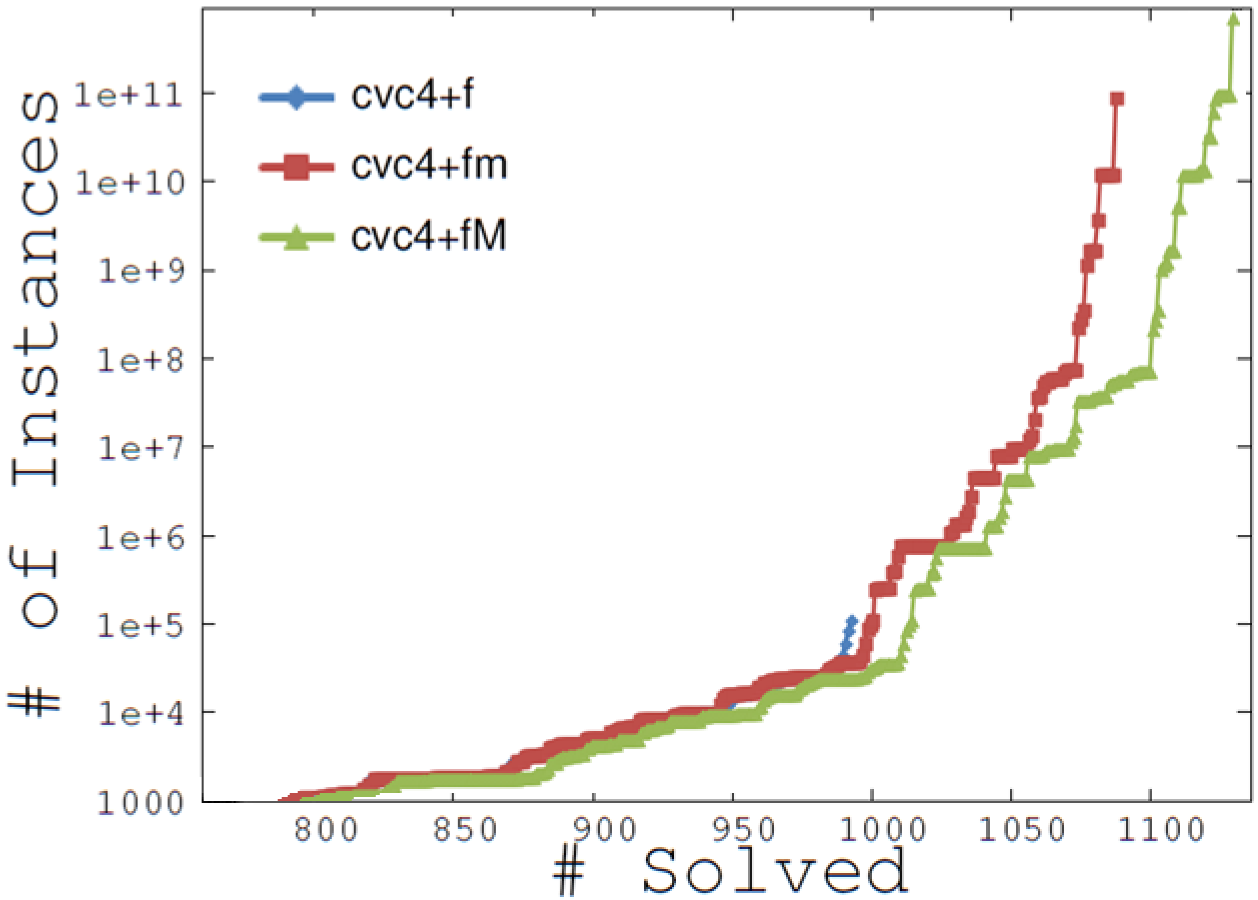
cvc4 :

- f : finite model
- i : heuristic
- m : model-based
- M : model-based (version 2)

10 second timeout

- CVC4 Placed 3rd in FNT (non-theorem) division of CASC 24

Results : TPTP



cvc4 :

- f : finite model
- m : model-based
- M : model-based (version 2)

- Model-Based Instantiation is often essential
  - Solves when naïve approach requires ~775 billion instances

Results: Isabelle

SAT	ArrowOrder	FFT	FTA	Hoare	NS_Shared	QEpres	StrongNorm	TwoSquares	TypeSafe	Total
cvc3	0	9	0	0	0	0	0	8	0	17
z3	1	19	24	46	10	47	1	17	12	177
cvc4+i	0	9	0	0	0	0	0	8	0	17
cvc4+f	26	123	**163**	149	56	75	12	**50**	84	738
cvc4+fi	26	133	158	**155**	**61**	80	12	44	**87**	756
cvc4+fm	22	120	152	147	36	75	12	46	87	697
cvc4+fM	28	126	163	151	44	94	12	43	87	748
cvc4+fMi	**31**	**136**	161	154	61	**101**	**12**	44	85	**785**

UNSAT	ArrowOrder	FFT	FTA	Hoare	NS_Shared	QEpres	StrongNorm	TwoSquares	TypeSafe	Total
cvc3	**287**	**250**	**877**	**577**	102	**291**	206	**552**	216	**3358**
z3	254	230	797	507	**135**	242	**240**	491	**329**	3225
cvc4+i	253	233	749	476	99	265	234	523	267	3099
cvc4+f	123	94	350	209	41	99	83	361	127	1487
cvc4+fi	155	164	509	374	37	168	100	452	195	2154
cvc4+fm	112	86	357	212	26	119	82	349	120	1463
cvc4+fM	88	92	381	202	29	109	93	365	149	1508
cvc4+fMi	154	164	515	371	37	167	100	452	195	2155

cvc4 :

- f : finite model
- i : heuristic
- m : model-based
- M : model-based (version 2)

10 second timeout

- For UNSAT, cvc4 with finite model finding is orthogonal :
  - Solves 170 unsat that cvc3 cannot, 365 z3 cannot, 229 that cvc4+i cannot

Extension to (Bounded) Integers

- A formula of the form

$\forall x_1 \ldots x_n : \text{Int}.\ L_1 \leq x_1 \leq U_1 \wedge \ldots \wedge L_n \leq x_n \leq U_n \Rightarrow \Psi$

- Where $x_i \notin FV(L_j, U_j)$, for $i < j$

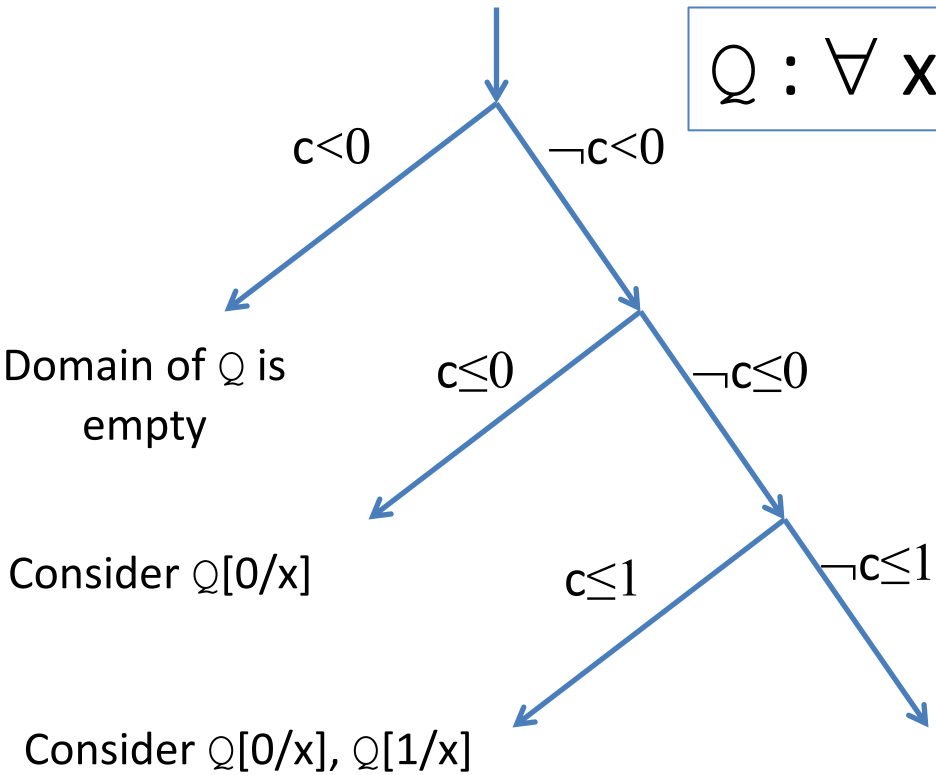
has *Bounded Integer Quantification*

- Example : $\forall xy.\ 0 \leq x \leq 20 \wedge 0 \leq y \leq f(x) \Rightarrow P(x, y)$
- Can be handled similar as before
  - Minimize bounds, (naïvely) instantiate exhaustively

Bounded Integer Quantification

- Idea: Fix values of bound c

$$Q : \forall\, x : \text{Int}.\ 0 \le x \le c \Rightarrow P(x)$$



- Approach is sound, and model complete
  - When input has model, it eventually terminates with “SAT”

Results

	SAT (263)		UNSAT (843)	
	solved	time	solved	time
z3	257	957.9	843	20.3
cvc4+i	0	0	**843**	17.4
cvc4+fi	**263**	90.8	843	308.7

cvc4 :

- f : bounded integer techniques
- i : heuristic

600 second timeout

- Set of verification benchmarks from Intel
  - Arrays, datatypes, integer arithmetic
  - Symbolic bounds for integer quantification, e.g.

    $\forall x : Int.\ 0 \leq x \leq c \Rightarrow P(x)$, where c is free constant
- CVC4 (with fmf) finds small models $\mathbb{M}$, i.e.
  - Value of $\mathbb{M}[c]$ is 2 to 3, at most 10

Summary

- CVC4 with finite model finding:
  - Incorporates various instantiation strategies:
    - Model-based quantifier instantiation
    - Heuristic instantiation (E-matching)
  - Has important properties:
    - Finite-Model Completeness
    - Refutational Completeness (under certain conditions)
  - Approach can be extended to integers, theory of strings
  - Improves the state-of-the-art, over:
    - SMT solvers
      - Increased ability to answer "satisfiable"
    - Automated Theorem Provers
      - Efficient reasoning about background theories at QF level

Thank you

- Acknowledgements:
  - Collaborators: Cesare Tinelli, Amit Goel, Sava Krstíc, Clark Barrett, Morgan Deters, Leonardo de Moura
  - Dissertation Committee: Aaron Stump, Hantao Zhang, Sriram Pemmaraju

- Questions?