Integrating Simplex with DPLL(T)

Reporter: Chi Zhiming

August 22, 2022

目录

目录

Contribution

- We have presented a new Simplex-based solver designed for efficiently solving SMT problems in solving linear arithmetic.
- The main features of the new approach include the possibility to presimplify the input problem by eliminating variables, a reduction in the number of slack variables, and fast backtracking.
- Another result of the paper is a simple approach for solving strict inequalities that does not require modification of the basic Simplex algorithm.

目录

DPLL(T) approach

If ϕ is a formula built by boolean combination of atoms of quantifier-free theory T, then the satisfiability of ϕ can be decided by combining a boolean satisfiability solver and a $T-$ *solver*.

- First Φ is transformed into a propositional formula Φ_0 by replacing its atoms $\phi_1, \ldots, \phi_t$ with fresh propositions $p_1, \ldots, p_t$
- A boolean valuation for Φ_0 is a mapping b from Φ_0 's propositions to $\{0, 1\}$. Any such b defines a set of atoms $\Gamma_b = \{\gamma_1, \ldots, \gamma_t\}$ where γ_i is ϕ_i if $b(p_i) = 1$ and γ_i is $\neg\phi_i$ if $b(p_i) = 0$.
- Then Φ is satisfiable if there is a b that satisfies Φ_0 (in propositional logic) and for which Γ_b is consistent (in theory T).

Solvers for DPLL(T): Terminology

- State:, a $T-$ *solver* maintains a **state** that is an internal representation of the atoms asserted so far.
  This solver must provide operations for updating the state by asserting new atoms, checking whether the state is consistent, and for backtracking.
- Theory propagation: identify atoms that are implied by the current state.
- Explanation:
  - In an inconsistent state S, an explanation is any inconsistent subset of the atoms asserted in S.
  - Similarly, an explanation for an implied atom γ is a subset Γ of the asserted atoms such that $\Gamma \models \gamma$.

  To interact with the DPLL search, the solver must produce explanations for conflicts and propagated atoms
- Minimal: In both above cases, the explanation is **minimal** if no proper subset of Γ is

General Settings

Initial:

- fixed formula ϕ.
- The set of atoms asserted so far is denoted by α.
- Stack of checkpoints: mark consistent states to which the solver can backtrack.

General Settings

API:

- *Assert(γ)*: asserts atom γ in the current state. It returns either ok or unsat $\langle\Gamma\rangle$ where Γ is a subset of α. In the first case, γ is inserted into α. In the latter case, $\alpha \cup \{\gamma\}$ is inconsistent and Γ is the explanation.
- *Check()*: checks whether α is consistent. If so, it returns ok, otherwise it returns unsat $\langle\Gamma\rangle$. As previously $\Gamma \subseteq \alpha$ is an explanation for the inconsistency. A new checkpoint is created when ok is returned.
- *Backtrack()*: backtracks to the consistent state represented by the checkpoint on the top of the stack.
- *Propagate()*: performs theory propagation. It returns a set $\{\langle\Gamma_1, \gamma_1\rangle, \ldots, \langle\Gamma_t, \gamma_t\rangle\}$ where $\Gamma_i \subseteq \alpha$ and $\gamma_i \in \mathcal{A}\backslash\alpha$. Every γ_i is implied by Γ_i

General Settings

- *Assert* must be sound but is not required to be complete: $Assert(\gamma)$ may return ok even if $\alpha \cup \{\gamma\}$ is inconsistent.
- *Propagate* must be sound but does not have to be exhaustive.A solver without theory propagation can return $Propagate() = \emptyset$ for any S
- *Check* is required to be sound and complete
- Several calls to *Assert* followed by a single call to *Check*.
- The state S' after executing Backtrack must be logically equivalent to the state S when the checkpoint was created, but S' may be different from S.

目录

Preprocessing

Rewriting: ϕ into an equisatisfiable formula of the form $\phi_A \wedge \phi'$

- ϕ_A is a conjunction of linear equalities
- ϕ' are **elementary atoms** of the form $y \bowtie b$, where y is a variable and b is a rational constant.

$$x \geq 0 \wedge (x + y \leq 2 \vee x + 2y - z \geq 6) \wedge (x + y = 2 \vee x + 2y - z > 4)$$

$$\Longleftrightarrow$$

$$(s_1 = x + y \wedge s_2 = x + 2y - z) \wedge$$
$$(x \geq 0 \wedge (s_1 \leq 2 \vee s_2 \geq 6) \wedge (s_1 = 2 \vee s_2 > 4))$$

Preprocessing

- Then ϕ_A can be written in matrix form as $Ax = 0$.
- checking the satisfiability of ϕ in linear arithmetic $\iff$ checking the satisfiability of ϕ' in **linear arithmetic modulo** $Ax = 0$.
- Solver: deciding the consistency of a set of elementary atoms Γ modulo the constraints $Ax = 0$
- If Γ contains only equalities and (non-strict) inequalities,this problem can reduce to search $x \in R$ for:

$$Ax = 0 \text{ and } l_j \leq x_j \leq u_j \text{ for } j = 1, \ldots, n \tag{1}$$

Preprocessing

- Reduce the matrix size: Simplify the costraints $Ax = 0$ by removing any variable x_i that does not occur in any elementary atom of ϕ' by Gaussian elimination.
- the presence of both lower and upper bounds is beneficial

Basic Solver

Initial:

- Tableau: From $Ax = 0$,

$$x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j \quad x_i \in \mathcal{B},$$

- β: assigns a rational value $\beta(x_i)$ to every variable x_i; $\beta_0(x_j) = 0$
- Bound: $l_j = -\infty, u_j = \infty$ for all j
- Invariants: $\forall x_j \in \mathcal{N}, l_j \leq \beta(x_j) \leq u_j$

Basic Solver

Two auxiliary procedures that modify β:

procedure update(x_i, v)
  for each $x_j \in \mathcal{B}$, $\beta(x_j) := \beta(x_j) + a_{ji}(v - \beta(x_i))$
  $\beta(x_i) := v$

procedure pivotAndUpdate(x_i, x_j, v)
  $\theta := \frac{v - \beta(x_i)}{a_{ij}}$
  $\beta(x_i) := v$
  $\beta(x_j) := \beta(x_j) + \theta$
  for each $x_k \in \mathcal{B} \setminus \{x_i\}$, $\beta(x_k) := \beta(x_k) + a_{kj}\theta$
  pivot(x_i, x_j)

Assertion Procedures

We use these two procedures to update the bounds l_i and u_i.

1. **procedure** AssertUpper($x_i \leq c_i$)
2. **if** $c_i \geq u_i$ **then return** *satisfiable*
3. **if** $c_i < l_i$ **then return** *unsatisfiable*
4. $u_i := c_i$
5. **if** x_i is a non-basic variable and $\beta(x_i) > c_i$ **then** update(x_i, c_i)
6. **return** *ok*

1. **procedure** AssertLower($x_i \geq c_i$)
2. **if** $c_i \leq l_i$ **then return** *satisfiable*
3. **if** $c_i > u_i$ **then return** *unsatisfiable*
4. $l_i := c_i$
5. **if** x_i is a non-basic variable and $\beta(x_i) < c_i$ **then** update(x_i, c_i)
6. **return** *ok*

Check

Modify the $\beta(x_i)$ for basic variable x_i such that $l_i \leq \beta(x_i) \leq u_i$

```
1. procedure Check()
2.   loop
3.     select the smallest basic variable x_i such that β(x_i) < l_i or β(x_i) > u_i
4.     if there is no such x_i then return satisfiable
5.     if β(x_i) < l_i then
6.       select the smallest non-basic variable x_j such that
7.         (a_ij > 0 and β(x_j) < u_j) or (a_ij < 0 and β(x_j) > l_j)
8.       if there is no such x_j then return unsatisfiable
9.       pivotAndUpdate(x_i, x_j, l_i)
10.    if β(x_i) > u_i then
11.      select the smallest non-basic variable x_j such that
12.        (a_ij < 0 and β(x_j) < u_j) or (a_ij > 0 and β(x_j) > l_j)
13.      if there is no such x_j then return unsatisfiable
14.      pivotAndUpdate(x_i, x_j, u_i)
```

Generating Explanations

An inconsistency may be detected by *Check* at line 8 or 13. Assume at line 8:

- Basic variable x_i such that $\beta(x_i) < l_i \implies a_{ij} > 0 \Rightarrow \beta(x_j) = u_j$ and $a_{ij} < 0 \Rightarrow \beta(x_j) = l_j$ for all non-basic variable x_j
- It follows that: Let $\mathcal{N}^+ = \{x_j \in \mathcal{N} \mid a_{ij} > 0\}$, Let $\mathcal{N}^- = \{x_j \in \mathcal{N} \mid a_{ij} < 0\}$

$$\beta(x_i) = \sum_{x_j \in \mathcal{N}} a_{ij}\beta(x_j) = \sum_{x_j \in \mathcal{N}^+} a_{ij}u_j + \sum_{x_j \in \mathcal{N}^-} a_{ij}l_j.$$

- And $x_i = \sum_{x_j \in \mathcal{N}} a_{ij}x_j$, we have:

$$\beta(x_i) - x_i = \sum_{x_j \in \mathcal{N}^+} a_{ij}(u_j - x_j) + \sum_{x_j \in \mathcal{N}^-} a_{ij}(l_j - x_j)$$

Generating Explanations

- Then one can derive the following implication:

$$\bigwedge_{x_j \in \mathcal{N}^+} x_j \leq u_j \wedge \bigwedge_{x_j \in \mathcal{N}^-} l_j \leq x_j \Rightarrow x_i \leq \beta(x_i)$$

- Since $\beta(x_i) < l_i$, this is inconsistent with $l_i \leq x_i$.
- The explanation for the conflict is then the following set of elementary atoms:

$$\Gamma = \{x_j \leq u_j \mid j \in \mathcal{N}^+\} \cup \{x_j \geq l_j \mid j \in \mathcal{N}^-\} \cup \{x_i \geq l_i\}.$$

- Γ is minimal.

Backtracking

- Save the value of $u_i(l_i)$ on a stack before it is updated by the procedure *AssertUpper (AssertLower)*.
- Only the assignment β corresponding to the last successful Check needs to be stored.
- The assignment β is a model for the current set of constraints and for the set of constraints asserted at any previous checkpoint.

Theory Propagation

- *unate propagation*: $x_i \geq c_i \implies x_i \geq c'$ with $c' < c_i$, is useful in **practice**.
- bound Refinement: For $x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$,use bounds of x_j to derive a lower or upper bound of x_i
- It can combine with any equality $a_1 x_1 + \ldots + a_n x_n = 0$ derived by linear combination of rows of A(**In NN?**)

Example

$A_0 = \begin{cases} s_1 = -x + y \\ s_2 = x + y \end{cases}$		$\beta_0 = (x \mapsto 0, y \mapsto 0, s_1 \mapsto 0, s_2 \mapsto 0)$
$A_1 = A_0$	$x \leq -4$	$\beta_1 = (x \mapsto -4, y \mapsto 0, s_1 \mapsto 4, s_2 \mapsto -4)$
$A_2 = A_1$	$-8 \leq x \leq -4$	$\beta_2 = \beta_1$
$A_3 = \begin{cases} y = x + s_1 \\ s_2 = 2x + s_1 \end{cases}$	$-8 \leq x \leq -4$ $s_1 \leq 1$	$\beta_3 = (x \mapsto -4, y \mapsto -3, s_1 \mapsto 1, s_2 \mapsto -7)$

目录

Conversion to non-strict Inequalities

引理

A set of linear arithmetic literals Γ *containing strict inequalities* $S = \{p_1 > 0, \ldots, p_n > 0\}$ *is satisfiable iff there exists a rational number* $\delta > 0$ *such that for all* δ' *such that* $0 < \delta' \leq \delta$, $\Gamma_\delta = (\Gamma \cup S_\delta) \setminus S$ *is satisfiable, where* $S_\delta = \{p_1 \geq \delta, \ldots, p_n \geq \delta\}$

Rather than computing an explicit value for δ, we treat it symbolically, as an *infinitesimal parameter*.

$\mathbb{Q}_\delta$

A pair (c, k) of $\mathbb{Q}_\delta$ is denoted by $c + k\delta$ and the following operations and comparison are defined in $\mathbb{Q}_\delta$:

$$(c_1, k_1) + (c_1, k_2) \equiv (c_1 + c_2, k_1 + k_2)$$
$$a \times (c, k) \equiv (a \times c, a \times k)$$
$$(c_1, k_1) \leq (c_2, k_2) \equiv (c_1 < c_2) \vee (c_1 = c_2 \wedge k_1 \leq k_2)$$

Conversion to non-strict Inequalities

And the transformation rules are such as:

$$
\begin{array}{lcll}
x \leqslant d & \mapsto & x \leqslant d + 0 \cdot \delta & \mapsto & x \leqslant \langle d, 0 \rangle \\
x < d & \mapsto & x \leqslant d + f \cdot \delta & \mapsto & x \leqslant \langle d, f \rangle \\
x \geqslant d & \mapsto & x \geqslant d + 0 \cdot \delta & \mapsto & x \geqslant \langle d, 0 \rangle \\
x > d & \mapsto & x \geqslant d + g \cdot \delta & \mapsto & x \geqslant \langle d, g \rangle \\
x = d & \mapsto & x = d + 0 \cdot \delta & \mapsto & x = \langle d, 0 \rangle \\
x \neq d & \mapsto & x \neq d + 0 \cdot \delta & \mapsto & x \neq \langle d, 0 \rangle
\end{array}
$$

where $f < 0, g > 0$

目录

Branch and Bound

For constraints:

$$\begin{aligned} Ax &= 0 \\ l_j \leq x_j \leq u_j &\text{ for } j = 1, \ldots, n \end{aligned} \tag{2}$$

- First,solving the linear programming relaxation to get $\beta(x_i)$.
- For $c < \beta(x_i) < c+1$ is not integer, the original problem S is split into the following two subproblems:

Branch and Bound

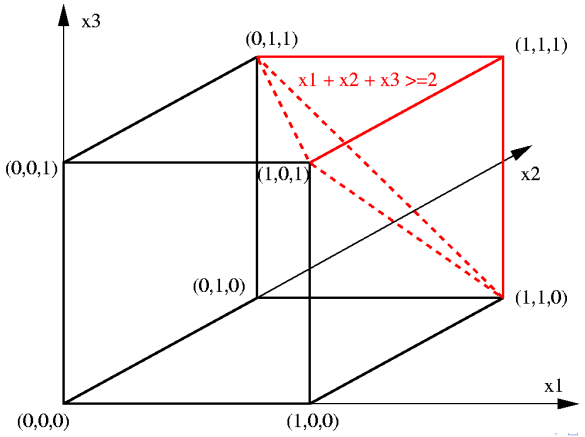
$$S_0: \begin{array}{rcl} Ax & = & 0 \\ l_j \leq & x_j & \leq u_j \quad \text{for } j = 1, \ldots, n \text{ and } j \neq i \\ l_i \leq & x_i & \leq c \end{array}$$

$$S_1: \begin{array}{rcl} Ax & = & 0 \\ l_j \leq & x_j & \leq u_j \quad \text{for } j = 1, \ldots, n \text{ and } j \neq i \\ c+1 \leq & x_i & \leq u_i. \end{array}$$

Gomory Cuts

- Assume β is a solution to the LP-relaxation P' of S but not a solution to S. A *cut* is a linear inequality:
$$a_1 x_1 + \ldots + a_n x_n \leq b$$
- If one or more such cuts can be found, then they can be added as new constraints to the original problem S. This gives a new problem S' that has the same solutions as S but whose LP-relaxation P' is strictly more constrained than P.

Gomory Cuts



目录

Experiments

	sat	unsat	failed	time (secs)
Ario 1.1	186	640	517	1218371
BarcelogicTools	153	417	92	401842
CVC Lite	117	454	772	1193747
MathSAT 3.3.1	330	779	234	739533
Yices	358	756	229	702129
Simplics	240	351	110	476940
New Solver	**412**	**869**	**62**	**267198**

- Efficient backtracking and the presimplification
- Efficient theory propagation based on bound refinement

Thank you