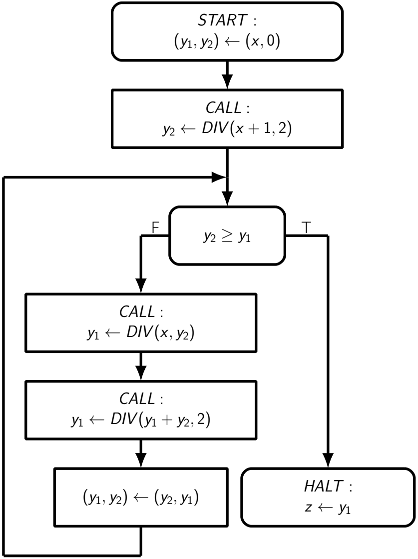
Лекция 5. Триггеры

Цель лекции

Научиться эффективно использовать SMT-солверы для дедуктивной верификации.

Содержание

Пример для дедуктивной верификации



$D_x = D_{y_1} = D_{y_2} = D_z = \mathbb{Z}$.
Надо доказать полную корректность этой блок-схемы относительно спецификации
$\varphi(x) = x \geq 0$,
$\psi(x,\ z) = (z^2 \leq x < (z+1)^2)$,
если сигнатуре DIV сопоставлена спецификация
$\varphi_D(a_1,\ a_2) = (a_1 \geq 0 \wedge a_2 > 0)$,
$\psi_D(a_1,\ a_2,\ r) = (0 \leq a_1 - r * a_2 < a_2)$.

Ход решения

- Докажите полную корректность этой блок-схемы «на листочке» самостоятельно.
- Выпишите условия верификации на языке Why3.
- Используйте Why3IDE для доказательства условий верификации при помощи солверов *Alt-Ergo*, *CVC4*, *Z3*.

Первые выводы

- Вы видите, что солверы не могут доказать некоторые из истинных условий верификации.
- Индуктивное утверждение можно записать с разными кванторами и вы видите, что с одними кванторами солвер доказывает больше условий верификации, чем с другими.
- Во всем этом надо разобраться.

Содержание

Солверы

- Для «доказательства условий верификации» мы используем *солверы*.
- Солвер – это алгоритм решения «уравнений» определенного вида (типовое «уравнение» – это формула со свободными переменными) или программа, реализующая этот алгоритм.
- Примеры: солвер дифференциальных уравнений, SAT-солвер, солвер тригонометрических уравнений, солвер логических уравнений.
- Солвер ожидает формулу на известном ему языке, т.е. с определенным набором операций. На них рассчитан алгоритм солвера. Их еще называют «теорией» (тригонометрия, логика и т.п.).

Солверы (2)

- У солвера есть 3 варианта ответа:
    - решение найдено, и оно предъявляется
    - солвер определил, что решений нет, формула противоречива
    - солвер не знает, какой из первых двух пунктов имеет место быть (не для всех уравнений ведь есть метод решения)

SMT-солвер

- SMT-солвер – это комбинация солверов. То есть это программа для решения «уравнений» из комбинации теорий.
- Как скомбинировать солверы? Например, получив формулу, можно сделать «замену переменных», как вы это делали при решении уравнений на вступительных экзаменах, и получить несколько задач, каждая из своей теории.
- Мы уже использовали теорию целых чисел.

Солвер как прувер

Как доказать истинность формулы (условия верификации) при помощи солвера?

- Условие верификации – это формула с квантором всеобщности (как минимум по значениям входных переменных).
- Надо взять ее отрицание и применить сколемизацию. Тогда получится формула со свободными переменными.
- Если солвер определит, что у этой формулы нет решений, то можно считать доказанным истинность исходной формулы.
- Если решение есть, то исходная формула противоречива и есть контрпример.
- Солвер может ответить, что не знает, есть ли решение.

Пруверы

- Альтернатива солверам – пруверы. Они проверяют данный пользователем вывод формулы в заданной логике. Вывод – это цепочка преобразований множества формул из аксиом. Допускаются только определенные преобразования («правила вывода»).
- Proof assistant – командная среда, помогающая пользователю составить вывод. Возможности варьируются: от полной автоматизации до лишь записи пошагового вывода, который полностью пишется пользователем.
- Главное: солверы – обычно более автоматические, чем пруверы. Поэтому мы используем солверы, а не пруверы.
- Примеры пруверов: *Coq*, *HOL/Isabelle*, *PVS*.

Содержание

Кванторы в солверах

- Пусть есть формула для условия верификации, истинность которой надо доказать. Берем отрицание формулы и сколемизируем ее. Получаем конъюнкцию множества подформул.
- Могут остаться кванторы всеобщности в конъюнктах. Что с ними делать? Подбирать значения подкванторных переменных и инстанцировать кванторы, чтобы получить противоречащие подформулы.

Пример с инстанцированием

- При помощи инстанцирования кванторов докажите противоречивость данного набора формул. Все свободные переменные принадлежат $\mathbb{Z}$. $\begin{cases} x > 0 \\ \forall t \in \mathbb{Z} \cdot t > 0 \Rightarrow x * t = x \end{cases}$
- Инстанцируем квантор значением $t = 2$. Получаем новую формулу $2 > 0 \Rightarrow x * 2 = x$. После упрощения: $x * 2 = x$. Целочисленный солвер это упростит до $x = 0$. Получили противоречие с $x > 0$.

Выводы из примера

- Как автоматически определить, что можно подставить 2 в качестве t? Множество значений переменной t бесконечно, его не перебрать.
- В общем случае, задача выбора подстановок, чтобы получить противоречащие формулы, алгоритмически неразрешима...
- Может быть вместо инстанцирования надо ввести дополнительные утверждения, доказать их и использовать? Так делает человек. Но солвер не умеет «придумывать» дополнительные утверждения.

Проблема инстанцирования кванторов всеобщности

- Если солвер не может придумывать, то он должен использовать то, что уже имеет, т.е. части самой формулы и части других инстансов кванторов.
- Бесконтрольное инстанцирование приводит к большому числу ненужных формул, причем рост может быть экспоненциальным.
- Нужна методика ограничивания инстанцирования.

Триггеры

- Триггер – это терм, в котором встречаются все подкванторные переменные как свободные термы триггера.
- Если солвер в процессе работы имеет терм, унифицирующийся с триггером, он определяет подстановку для подкванторных переменных квантора и инстанцирует квантор.

Варианты триггеров

- Мультитриггер – это множество термов, в совокупности дающих подстановку всех подкванторных переменных.
- Может быть несколько триггеров у одного квантора. Если можно провести унификацию хотя бы с одним из них, она производится.

Пример

- Пример квантора с триггером (триггер записан в квадратных скобках после подкванторных переменных): `forall x y [f x y]. (g x) -> (f (h y) x)`
- Терм `f 0 1` унифицируется с триггером и дает такой инстанс квантора: `(g 0) -> (f (h 1) 0)`.
- В этом инстансе есть терм, который можно унифицировать с триггером. Это терм `f (h 1) 0`. Получаем новый инстанс квантора: `(g (h 1)) -> (f (h 0) (h 1))`.
- Опять можно инстанцировать аксиому...

Бесконечный цикл инстанцирования

- В этом примере можно попасть в бесконечный цикл инстанцирования квантора. Это еще одна причина, почему солвер может давать ответ «не знаю».
- Чтобы не попадать в этот цикл, надо выбрать другой триггер или вместо исходного квантора записать другой, эквивалентный ему. Например, триггер `f (h y) x`. Тогда при инстанцировании этого квантора не появится новых термов, унифицируемых с триггером.
- Другая проблема – слишком требовательный триггер. Солвер может никогда не получить нужный терм для унификации с триггером.

Генерация триггеров

- Если триггер не указан, солвер обычно выбирает его на основе некоторой эвристики.
- Пример эвристики выбора триггера:
    1. это терм, являющийся частью самого квантора;
    2. в терме встречаются все подкванторные переменные;
    3. в терм входит хотя бы один пользовательский символ;
    4. это минимальный терм (в нем самом нельзя выделить терм, в который входят все подкванторные формулы).
- Такую эвристику используют солверы *Alt-Ergo*, *CVC3*.

Пример

- Например, для квантора `forall x y. (x > y) -> (f (h y) x)` таким термом является `(f (h y) x)`.
- Терм `forall x y. (x > y) -> (f (h y) x)` не является минимальным.
- В терм `x > y` не входит пользовательский символ.
- В терм `(h y)` не входят все подкванторные переменные.

Отладка автоматического выбора триггера

- Солвер *Alt-Ergo* позволяет понять, какой триггер он выбрал для квантора и как он его использовал.
- `alt-ergo -dtriggers input.why` распечатывает автоматически сгенерированные триггеры
- `alt-ergo -dmatching=1 input.why` распечатывает кванторы, которые были инстанцированы (вместо 1 можно написать 2 для более подробного вывода)
- Ввод для *Alt-Ergo* можно получить из Why3IDE: кнопка Edit по попытке доказательства этим солвером и текст будет создан в файле в поддиректории с вспомогательными файлами.

Пример с подстановкой $t = 2$

- Ранее в этой лекции был пример системы формул, где противоречие находилось подстановкой $t = 2$. Что будет делать солвер с такой системой?
- Он выделит триггер $x * t$. Но термов для унификации с триггером нет. Значит солвер не сможет инстанцировать квантор. *Alt-Ergo* добавляет в систему еще один квантор (одно утверждение про умножение целых чисел) и он только и делает, что инстанцирует этот квантор. Его ответ – «не знаю».
- Правда, это не означает, что не может быть других эвристик: солверы *CVC4* и *Z3* находят противоречие.

Как доказать?

- Попробуем добиться, чтобы *Alt-Ergo* тоже находил противоречие. Надо добавить истинную формулу в систему формул, которая будет давать терм `x * 2`. Например, `x * 2 = x + x`.
- Теперь *Alt-Ergo* мгновенно находит противоречие.

Выводы

- Солверы могут давать ответ «не знаю» или зацикливаться, если им дают формулы с кванторами всеобщности.
- Если солвер зациклился или «не знает» ответ, посмотрите, надо ли ему работать с кванторами всеобщности. Какие триггеры выбрал солвер для кванторов всеобщности. Какие он должен сделать подстановки для подкванторных переменных. Достаточно ли солверу термов, чтобы сделать эти подстановки.
- Теперь вы можете ответить, почему в примере с дедуктивной верификацией квадратного корня солверы доказывают больше или меньше условий верификации в зависимости от того, как записано индуктивное утверждение.