Верификация программ и темпоральные логики

Лекция N 3 курса “Современные задачи теоретической информатики”

Юрий Лифшиц
yura@logic.pdmi.ras.ru

ИТМО

Осень’2005

"Если бы строители строили здания так же, как программисты пишут программы, первый залетевший дятел разрушил бы цивилизацию"

Второй закон Вейлера

План лекции

План лекции

Мотивация

В каких технологиях очень высока цена ошибки?

- Управление транспортом
- Медицинские системы
- Электронный бизнес
- Телефонные сети

Классические примеры:

Ракета Ariane-5 — ущерб 1000000000$
Медицинский ускоритель Therac-25 — 6 смертельных исходов

Детали — `http://www.softwarer.ru/safety.html`

Методы поиска ошибок

Четыре основных подхода:

- Имитационное моделирование (т.е. тестирование прототипа)
- Тестирование (полной программы)
- Дедуктивный анализ
- Верификация модели программы

Стадии верификации модели

Моделирование

Например, во время компиляции
Часто абстрагируются от неважных деталей
Трудность: не потерять значимые детали

Спецификация

Трудность: сформулировать исчерпывающие требования к программе

Верификация модели

Анализ контрпримеров
Если алгоритм не справляется — уменьшаем модель
Бывают “ложные опровержения” — ошибки в моделях, но не в программе $\Rightarrow$ нужно менять модель

Основные инструменты

Моделирование
Модель Крипке

Спецификация
Темпоральные логики: CTL, CTL*, LTL
μ-исчисление [на 5-ой лекции]

Верификация
Символьные алгоритмы [на 4-ой лекции]
Использование специальных структур данных (OBDD)
Редукция частичных порядков

План лекции

Неформально о моделировании

Два типа систем:
Одноразовый запуск. Проверятся Input-Output поведение
Реагирующая система, бесконечное время работы

Модель реагирующей системы:
Состояния
Возможные переходы

Атомарность переходов
Слишком большие — можем пропустить ошибку
Слишком маленькие — добавляем состояния, которые не достижимы на практике

Модель Крипке

AP — множество атомарных высказываний. Модель Крипке над AP — четверка $M = (S, S_0, R, L)$, в которой:

1. S - конечное множество состояний
2. $S_0 \subseteq S$ — множество начальных состояний
3. $R \subseteq S \times S$ отношение переходов
4. $L : S \to 2^{AP}$ — функция истинности

Последовательность $\pi = s_0 s_1 \ldots$ — путь в модели Крипке из состояния s, если $s_0 = s$ и для всех i выполнено $R(s_i, s_{i+1})$.

Крипке и другие модели

К модели Крипке могут быть сведены многие представления программ:

- Представление состояний и переходов логической формулой
- Булевы (логические) схемы
- Последовательные и параллельные программы

Темпоральные логики

Неформально, темпоральная логика — это язык, на котором можно формулировать утверждения, используя понятие времени.

Пусть есть набор переменных, которые как-то меняются со временем. Темпоральная логика позволяет формулировать утверждения типа:

- Значение a все время будет равно значению b
- Наступит момент, когда c станет нулем
- Значение d будет становится единицей бесконечно много раз

Синтаксис и семантика

Синтаксис
Правила составления формальных выражений

Семантика
Правила интерпретации формальных выражений

Кванторы пути и темпоральные операто

Кванторы пути:

- **A** — “выполнено для всех путей”
- **E** — “для некоторого пути”

Темпоральные операторы:

- **X** — “в следующий момент”
- **G** — “когда-нибудь, рано или поздно”
- **F** — “всегда, повсюду”
- **U** — “когда-нибудь наступит утв.2, а до него все время будет утв.1”
- **R** — “утв.2 будет выполнено до тех пор пока не появится утв.1”

Синтаксис CTL*

Пусть AP — атомарные высказывания

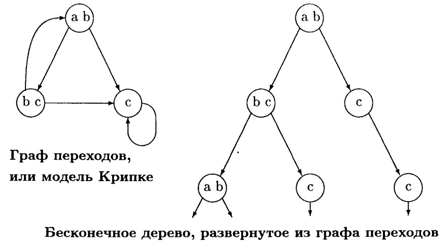
Синтаксис формул состояния:

- Если $p \in AP$, то p — формула состояния
- Если f и g — ф.с., то $\neg f$, $f \vee g$, $f \wedge g$ — ф.с.
- Если f — формула пути, то $\mathbf{A}f$ и $\mathbf{E}f$ — формулы состояния

Синтаксис формул пути:

- Если f — формула состояния, то f — формула пути
- Если f и g — формулы пути, то $\neg f$, $f \vee g$, $f \wedge g$, $\mathbf{X}f$, $\mathbf{G}f$, $\mathbf{F}f$, $f\mathbf{U}g$, $f\mathbf{R}g$ — формулы пути

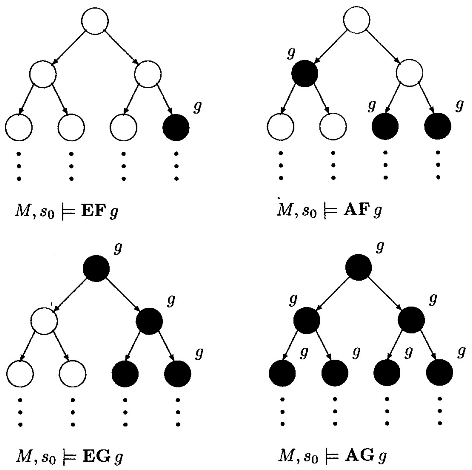
Семантика CTL* на модели Крипке



Обозначение $\boxed{M, s \models f}$: формула состояния f выполнена на модели M со стартовой вершиной s.

Отношение $\models$ определяется естественным образом индукцией по строению формулы.

Примеры



План лекции

CTL

Логика CTL

Сужение CTL*, допускающая только конструкции вида:

- $\neg f$
- $f \vee g$
- $\mathbf{EX}f$
- $\mathbf{EG}f$
- $\mathbf{E}[f\mathbf{U}g]$

Проблема верификации моделей

Данные:

Модель Крипке $M = (S, R, L)$

Формула темпоральной логики f

Найти:

Множество $\{s \in S \quad | \quad M, s \models f\}$

Верификация для CTL

Идеи алгоритма

Выписать все подформулы состояния f

Для каждого состояния $s \in S$ создать список выполненных подформул

Вести построение "индукцией по построению f"

Простые случаи

- Нам уже даны выполняющие множества для атомарных формул
- Знаем выполняющее множество для $f \Rightarrow$ построим и для $\neg f$
- Знаем выполняющие множества для f и $g \Rightarrow$ построим и для $f \vee g$
- Сделаем один шаг назад от выполняющего множества f — получим выполняющее множество для $\mathbf{EX}f$
- $\mathbf{E}[f\mathbf{E}g]$ — отмечаем все g и строим деревья обратных путей вдоль f-вершин

Оператор EG

Строим выполняющее множество для **EG**f

- Выкидываем вершины, не выполняющие f
- Находим компоненты сильной связности [Алгоритм Тарьяна]
- Строим обратные деревья от этих компонент

Трудоемкость итогового алгоритма: $O(|f|(|S| + |R|))$

Задачка на дом

Найти минимальный набор операторов в CTL*, через который можно выразить все остальные

Последний слайд

Если не запомните ничего другого:

- Верификация на модели состоит из трех этапов: моделирования, спецификации и верификации
- Модели Крипке используются для описания программ, а темпоральные логики для описания требований к ним
- Основной проблемой для построения эффективных алгоритмов верификации является экспоненциальное число состояний в модели Крипке

Вопросы?