Dynamic Logic

Владимир Гладштейн, Андрей Заварин

Динамическая логика

Формальная система, позволяющая рассуждать о программах. В первую очередь — формально записывать спецификацию и доказывать соответствие ей данной программы. Также с её помощью можно доказывать эквивалентность двух программ, сравнивать выразительность различных конструкций языка программирования и др.

Динамическая логика

Отличие от классической логики: в последней истинность формулы ϕ *статична*. Она определяется лишь означиванием свободных переменных, входящих в ϕ. При этом конкретное означивание подразумевается зафиксированным. В динамической логике на уровне синтаксиса определены программы, чей основной смысл заключается в изменении значений переменных, которое влечет изменение истинности тех или иных формул.

Динамическая логика

Пример

$x := x + 1$ изменяет истинность формулы "x – чётный".

Программы и состояния

Программа – алгоритм, записанный на формальном языке. Переменные в программах могут принимать значения из определенного domain of computation – структуры, в сигнатуру которой входят некоторые константы $(0, 1,$ etc$)$, операции $(+, -, *)$ и проверки или тесты $(=, <)$, которые можно проводить над значениями переменных.

Пример

Пусть домен вычислений – $(\mathbb{Z}; +, *; 0, 1; <)$ со стандартными операциями. Тогда можно написать такую (довольно бессмысленную) программу:

```
if x + 1 < y then
    y := 0;
else
    x := x * y;
```

Программы и состояния

Программы обычно изменяют значения переменных. Формально мы определяем состояние: функцию, сопоставляющую каждой переменной некоторое значение. В каждый момент исполнения программы она находится в некотором состоянии. Команды это состояние изменяют.

Недетерминизм

ДЛ позволяет работать с недетерминированными программами. Например, можно рассматривать оператор $x := ?$, означающий "присвоить x произвольное значение". Этим значением может быть какая-то информация из внешнего мира. В дальнейшем мы подробнее обсудим недетерминизм, но эта особенность ДЛ приводит к следующему:

Модальность

ДЛ – модальная логика. Для рассуждений о том, как выполнение программы α влияет на истинность формулы ϕ, в ДЛ вводятся две модальные конструкции:

- $\langle\alpha\rangle\phi$ – программа α, запущенная из текущего состояния, *может* завершиться в состоянии, удовлетворяющем ϕ
- $[\alpha]\phi$ – двойственная конструкция. Программа α *обязана* завершиться в состоянии, удовлетворяющем ϕ.

Отношение входов и выходов

Для программы α на множестве всех состояний можно ввести бинарное отношение $R_\alpha = \{(s, t) \mid$ начав в состоянии s, α может завершиться в состоянии t$\}$. Это отношение составляет основу будущей семантики ДЛ.

Построение программ

Программы строятся индуктивно, начиная с некоторых примитивных (atomic) программ и тестов, с помощью операторов над программами (условные операторы, композиция и другие). В пропозициональной версии ДЛ примитивные программы – это просто символы $a, b, \ldots$ из некоторого счетного алфавита, как и примитивные тесты $(p, q, \ldots)$

Операторы над программами

Пример

Пусть α и β – программы, ϕ – проверка. Тогда можно получить новые программы:

- $\alpha; \beta$ – последовательное выполнение
- **if** ϕ **then** α **else** β
- **while** ϕ **do** α

Такие программы называются **while** программами. В дальнейшем мы будем работать с более общим классом *регулярных* программ.

Регулярные программы

Помимо большей общности, они хороши простыми по сравнению с **while** и **if** программными конструкциями. Определяются они следующим образом:

Регулярные программы

- любая примитивная программа является (регулярной) программой.
- если ϕ – проверка, то $\phi?$ – программа.
- если α и β – программы, то $\alpha; \beta$ – программа.
- если α и β – программы, то $\alpha \cup \beta$ – программа.
- если α – программа, то и α^* – программа.

Регулярные программы

- примитивные программы неделимы и выполняются за один шаг
- $\phi?$ проверяет условие ϕ. В случае успеха продолжает выполнение, в противном случае зацикливается.
- $\alpha; \beta$ – просто последовательное выполнение двух программ

Регулярные программы

- $\alpha \cup \beta$ – недетерминированный выбор одной из ветви вычислений и её выполнение.
- α^* – выполнить α недетерминированно выбранное конечное число раз (возможно, 0)

Регулярные программы

Пример

Операторы **while** программ можно выразить через регулярные:

- **if** ϕ **then** α **else** $\beta \stackrel{\text{def}}{=} (\phi?;\alpha) \cup (\neg\phi?;\beta)$
- **while** ϕ **do** $\alpha \stackrel{\text{def}}{=} (\phi?;\alpha)^*;\neg\phi?$

Регулярные программы

Логика Хоара

В PDL можно выразить "тройки Хоара"

Definition

Тройка Хоара $\{\phi\}\ \alpha\ \{\psi\} \stackrel{\text{def}}{=} \phi \to [\alpha]\psi$

Логика Хоара

Логика Хоара

$\{1 \leq M \wedge 1 \leq N \ \wedge M = m \wedge N = n\}$

while $M <> N$ do

    if M < N

        then N := M - N;

        else M := N - M;

    fi

od

$\{M = gcd(m, n)\}$

Проблемы

Coq — тотальный язык, а значит на нем, без дополнительных плагинов, можно написать алгоритм, терминируемость, которого, "очевидна" для Coq.

```
Fixpoint insert x t :=
  if t is Node l y r then
    if x < y then Node (insert x l) y r
    else if x > y then Node l y (insert x r)
    else Node l y r
  else Node Emp x Emp.
```

Ссылки

- Репозиторий
- A. Celik, K. Palmskog, M. Parovic, E. Jesús Gallego Arias, and M. Gligoric.
  Mutation Analysis for Coq.
  ASE 2019, San Diego, USA, November 2019.
- ssrnatlia
- Equations