Математическая логика и логическое программирование

mk.cs.msu.ru → Лекционные курсы
→ Математическая логика и логическое программирование (3-й поток)

Блок 55

Проверка правильности распределённых систем
Пара слов о методе model checking

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2022/2023, осенний семестр

Проверка правильности распределённых систем

Начнём с примера: рассмотрим две процедуры в синтаксисе языка C:

```
void стипендия() {
  счёт += 1 000;
}
```

```
void надбавка() {
  счёт += 1 000 000;
}
```

Если они выполняются последовательно в каком-либо порядке,
то можно легко убедиться (например, при помощи логики Хоара),
что стипендия с надбавкой начисляются корректно:

$$\{\text{счёт} = \text{x}\}$$
$$\text{счёт} := \text{счёт} + 1\,000;$$
$$\text{счёт} := \text{счёт} + 1\,000\,000;$$
$$\{\text{счёт} = \text{x} + 1\,001\,000\}$$

$$\{\text{счёт} = \text{x}\}$$
$$\text{счёт} := \text{счёт} + 1\,000\,000;$$
$$\text{счёт} := \text{счёт} + 1\,000;$$
$$\{\text{счёт} = \text{x} + 1\,001\,000\}$$

Проверка правильности распределённых систем

Начнём с примера: рассмотрим две процедуры в синтаксисе языка C:

```
void стипендия() {
  счёт += 1 000;
}
```

```
void надбавка() {
  счёт += 1 000 000;
}
```

На практике каждое из присваиваний может выполняться
и за несколько шагов (действий) — например:

1. Чтение значения аргумента-переменной
2. Прибавление константы к прочитанному значению
3. Запись результата в переменную

Для последовательно выполняющихся процедур
степень детализации семантики неважна, и ответ о корректности,
полученный при помощи логики Хоара, можно считать правильным

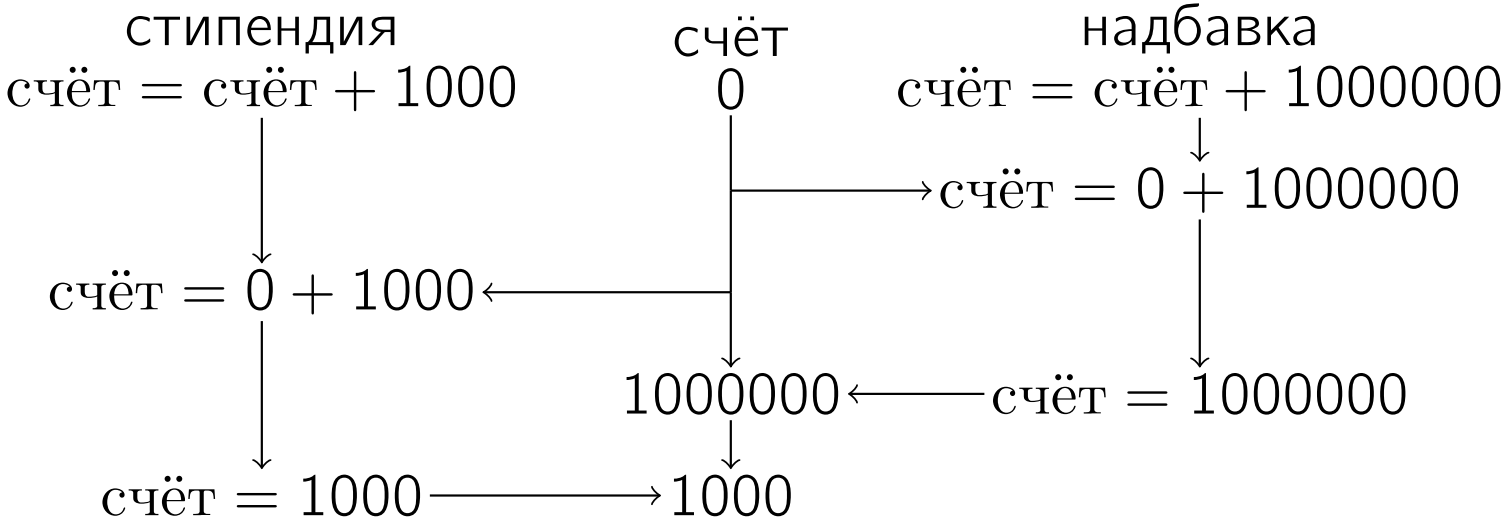
Проверка правильности распределённых систем

Начнём с примера: рассмотрим две процедуры в синтаксисе языка C:

```
void стипендия() {
  счёт += 1 000;
}
```

```
void надбавка() {
  счёт += 1 000 000;
}
```

Но если эти процедуры будут выполняться **параллельно**,
то ответ, полученный при помощи логики Хоара,
окажется не соответствующим реальности:



Честно заработанная надбавка исчезла со счёта
из-за неудачного редкого стечения технических обстоятельств

Проверка правильности распределённых систем

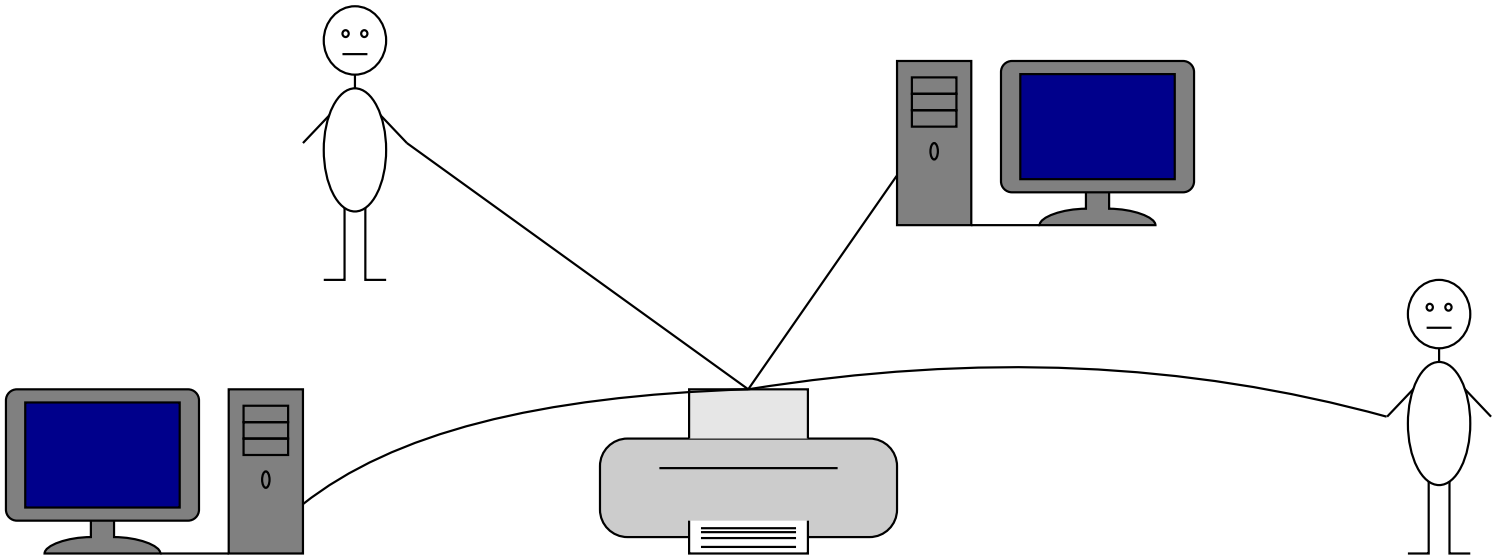
Такую ошибку нетрудно «проглядеть» при разработке программы

При этом её можно считать невоспроизводимой
и практически необнаружимой при помощи тестирования:

- Чтобы ошибка возникла, присваивания должны выполниться *почти одновременно* — настолько, чтобы третий шаг одного из них выполнился строго между первым и третьим шагами другого
- Разработчик программы обычно не может контролировать время выполнения шагов присваиваний настолько точно, чтобы можно было перемешивать шаги заранее заданным способом
- Даже если есть подходящие средства контроля точности, параллельно выполняющихся действий обычно настолько много, что нельзя считать разумным тестовое покрытие, содержащее всевозможные порядки выполнения
  - Например, для 70-ти параллельных независимых действий существует 70! порядков их выполнения, а это больше чем гугол

Проверка правильности распределённых систем

Ещё один пример



Представим себе сетевой принтер, с которым могут взаимодействовать пользователи при помощи удалённых компьютеров и находясь непосредственно у принтера

Как протестировать такую систему?

Можно ли, как-нибудь «разумно» протестировав систему, заключить, что в ней нет критичных ошибок?

Как могут быть устроены ошибки в такой системе?

Проверка правильности распределённых систем

Для проверки правильности распределённых систем, то есть таких, в которых компоненты выполняются параллельно, взаимодействуя между собой для достижения общей цели, одного только тестирования оказывается недостаточно:

- Достаточно полное тестирование зачастую чересчур трудоёмко или даже невозможно
- Протестировав систему по частям, как правило, нельзя быть уверенным в том, что вся система будет работать верно
- Некоторые критичные ошибки в работе системы возникают при настолько специфичных обстоятельствах, что обнаружить их тестированием практически невозможно

Поэтому для полноценной проверки правильности распределённых систем следует уметь применять и другие подходы — например, формальную верификацию

Пара слов о методе model checking

Для формальной верификации распределённых систем успешно применяется метод model checking
(проверка моделей; верификация моделей программ)

Согласно этому методу:

1. Описывается модель системы,
   - достаточно детальная, чтобы свойства модели можно было перенести на исследуемую систему, и при этом
   - достаточно простая для эффективного построения и анализа
2. Выбирается язык спецификаций,
   - соответствующий устройству модели и при этом
   - достаточно выразительный для записи желаемых требований
3. Разработчик создаёт и пользователи применяют средства **автоматической** проверки того, что модель соответствует её спецификации

Напоследок обсу́дим одну из «базовых» конкретных постановок задачи model checking и решение этой задачи

Пара слов о методе model checking

Краткая схема применения model checking