Математическая логика

mk.cs.msu.ru → Лекционные курсы → Математическая логика (318, 319/2, 241, 242)

Блок 45

Что ещё интересного есть в логике
Формальная верификация

Лектор:
Подымов Владислав Васильевич

E-mail:
valdus@yandex.ru

ВМК МГУ, 2020/2021, весенний семестр

Что ещё интересного есть в логике

Знакомство с математической логикой в этом курсе началось с изучения **логики высказываний**, **логики предикатов** и **проблемы общезначимости формул**

На примере **натуральных исчислений**, были показаны основы **теории доказательств** — одного из фундаментальных разделов логики

В разделе, посвящённом tbfаксиоматическим теориям, было показано, как можно применять логические методы для анализа высказываний о *нелогических* понятиях

В **модальных логиках** было показано, как вводить в язык “непривычные” операции, получая **неклассические** расширения логики

А что ещё интересного бывает в логике?

Что ещё интересного есть в логике

Из того, что не будет обсуждаться:

- × В языках **логик высшего порядка** кванторы разрешено применять к отношениям, семействам отношений, ...
- × В некоторых логиках (например, **интуиционистской**) “привычные” логические операции имеют совсем не такой смысл, как в логиках высказываний и предикатов

Из того, что успеем обсудить напоследок:

- ✓ Для решения некоторых прикладных задач, изначально, *казалось бы*, не связанных с логикой, успешно применяются логические методы

Формальная верификация

“Любая нетривиальная программа содержит хотя бы одну ошибку” (*автор неизвестен*)

А что такое “ошибка”, и насколько плохо, если она есть в программе?

Правильная программа не содержит ошибок и решает задачу, которую требовалось решить при помощи этой программы

А как убедиться, что заданная программа действительно решает поставленную задачу?

Формальная верификация

Подход к проверке правильности работы программы, про который знает каждый программист — это **тестирование**:

- придумывается показательный набор входных данных программы (тестовое покрытие)
- программа выполняется на тестовом покрытии
- результат выполнения программы сравнивается с результатом, ожидаемым согласно решаемой задаче

Основной недостаток такого подхода: **ошибки всё равно остаются** в программе, хотя их и становится меньше

Кроме того, в современном мире широко применяются вычислительные системы, для которых даже после "качественного" тестирования **не гарантируется отсутствие критичных ошибок**: сложные программные комплексы, распределённые системы, аппаратные и программно-аппаратные системы, интерактивные системы, ...

Формальная верификация

В некоторых случаях для приемлемого уровня правильности достаточно протестировать код
(“*более-менее работает, а дальше пусть пользователь разбирается*”),
но далеко не всегда: даже от небольших программных ошибок серьёзно зависит прибыль компаний[1], успешность проектов мирового масштаба[2], быт людей[3] и даже их жизни[4]

---

[1] 1994. Процессор Intel, ошибка в реализации деления чисел с плавающей точкой ⇒ замена дефектных процессоров, сотни миллионов $ убытка

[2] 1962–сейчас. Ракеты и спутники, взорвавшиеся и исчезнувшие из-за программных ошибок: Фобос (пропущена кавычка), Ariane 5 (ошибка в округлении чисел), Mars Global Surveyor (перепутаны английская и метрическая системы мер), Hitomi (ошибка в программе стабилизации вращения), ...

[3] 2003. Полное отключение электричества в нескольких областях США и Канады ⇐ ошибка в реализации взаимодействия программ оповещения об электрической нагрузке (race condition)

[4] 1980-е гг. Пять смертей при лечении рака аппаратом Therac-25 ⇐ ошибочное увеличение мощности радиационного облучения при очень редком сочетании времён выполнения параллельных подпрограмм (race condition)

Формальная верификация

Есть и дугой подход к проверке правильности вычислительных систем:

- формулируется набор требований к выполнению системы, означающих “система функционирует правильно”
- **строго** доказывается или опровергается утверждение о том, что система удовлетворяют этим требованиям

Требования такого вида, записанные на математическом языке, называются формальной спецификацей системы

Проверка соблюдения требований с помощью математических методов называется формальной верификацией системы

Если в качестве языка спецификации выбрать какой-либо **логический язык**, то для проверки правильности программы можно будет использовать **логические методы**