# Лекция 13. Моделирование Java-программ

# Цель лекции

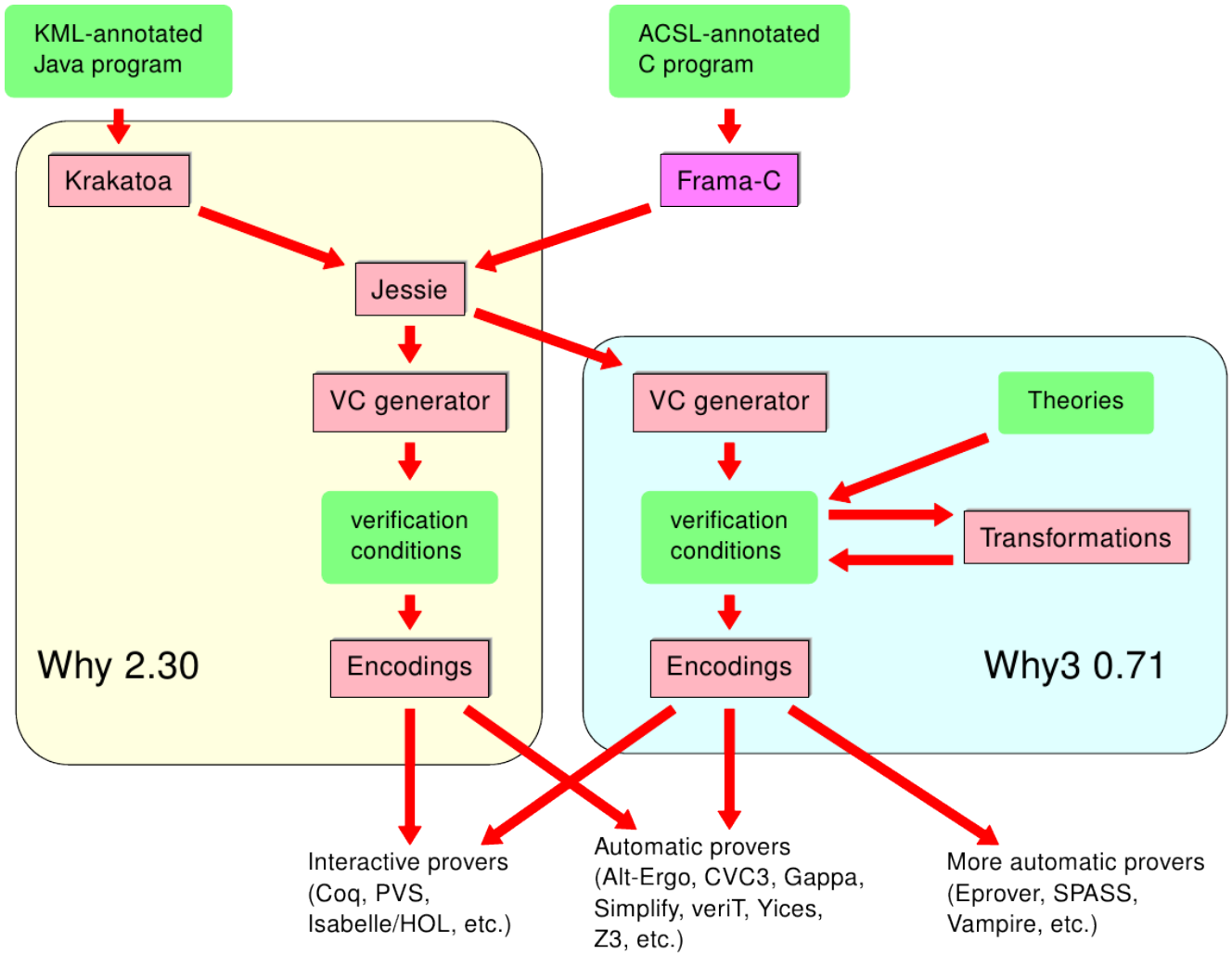
Обсудить моделирование Java-программы (в первую очередь, модель памяти), потому что AstraVer работает на той же аксиоматизации

# Содержание

# Jessie и Krakatoa

- AstraVer – наследник инструмента Jessie. Jessie моделирует Си-программу на Why3 и использует общую модель памяти с инструментом Krakatoa, который моделирует Java-программы.
- Krakatoa должен был доказывать правильность программ для JavaCard. После окончания проекта инструмент не развивается.

# Картинка с семейством инструментов



# Почему для Си модель памяти языка Java?

- Много похожего, хотя есть и отличия
- Можно переиспользовать готовые инструменты, модели, модули для Java, раз они уже были к моменту создания Jessie

# Содержание

# Чего нет в Си?

- Инкапсуляция (классы, поля, методы)
- Наследование
- Полиморфизм (виртуальные методы)
- Специфичные возможности: исключения, instanceof и приведения типа, оператор new, интерфейсы и т.п.
- Ссылочная модель памяти

# Модель требований

- Язык спецификации ООП-программы – это зыбкая область. Много вопросов, нет решений, принятых всем сообществом программистов.
- Основная проблема спецификаций ООП-программы – большое число требований к коду, потому что это ООП-код (с классами, с интерфейсами).
- Решение №1: в языке спецификации уже есть бОльшая часть моделей типовых требований; какие плюсы и минусы?
- Решение №2: язык спецификации содержит небольшое число возможностей и средства определения новых; какие плюсы и минусы?

# Модель требований у нас

- Ввиду отсутствия хороших решений, мы не будем касаться темы языка спецификации ООП-программ
- Будем предполагать, что каждый метод каждого класса снабжен парой из предусловия и постусловия

# Дополнение модели требований из-за исключений

- Т.к. метод может завершиться из-за необработанного исключения, то надо предусмотреть постусловие «для завершения по исключению»
- Это как будто мы вводим в блок-схему новый вид операторов (HALT-RAISES): оператор завершения блок-схемы из-за исключения
- Частичная корректность:
  1. если вычисление завершилось на операторе HALT-RAISES, то должно быть выполнено постусловие «для завершения по исключению»
  2. если вычисление завершилось на операторе HALT, то должно быть выполнено обычное постусловие

# Содержание

# Что надо уметь моделировать?

- обращение к полю (возможно, поле находится в классе-предке)
  - надо собрать поля из всех классов-предков
- вызов виртуального метода
  - предполагаем, что мы верифицируем фиксированную программу, с фиксированным набором классов и наследования; значит, можно перебрать динамические типы объекта в виде условного выбора и для каждого типа вызвать метод из нужного класса
  - нужно поддерживать динамические типы для каждого указателя
- генерация исключения и обработка исключения в разных методах
  - используем встроенные возможности Why3 по поддержке исключений

# Что надо уметь моделировать?

- оператор new
  - добавляем новый блок в alloc_table и вызываем метод-конструктор
- оператор instanceof
  - динамические типы уже поддерживаются, остается добавить соотношения на типах, что они находятся в иерархии наследования
- приведение типов ссылок
  - это всего лишь проверка динамического типа, сама ссылка и объект не меняются

# Упражнение

- Попробуйте применить эти идеи для доказательства полной корректности кода классов Account, CreditAccount и Prog (который тестирует предыдущие классы)

# Выводы

- Нужно добавлять поля класса-предка в класс-потомок
- Нужно моделировать иерархии классов по наследованию (tag – элемент иерархии, на тэгах определено отношение subtag, в иерархии есть нижний тип bottomtag).
- Модель памяти (из memory и alloc_table) нужно дополнить таблицей с динамическими типами переменных (tag_table)
- Тип переменной memory должен типизироваться типом bottomtag, т.к. в этой переменной должны быть указатели разных типов в рамках одной иерархии (т.к. типы указатели на класс-предок и на класс-потомок не отличаются, иначе было бы сложно делать приведение типов и обращение к полю класса-предка как приведения к нужному типу-предку и обращению к memory с его типом)