Сохранение скобок и описание синтаксиса языков

Кирилл Елагин

10 сентября 2015

Задача

- состоит из двух частей
- скорее теоретическая, но код писать тоже придется
- нужно подумать и применить фантазию

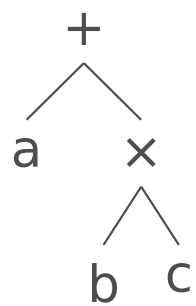
Первая часть: сохранение скобок

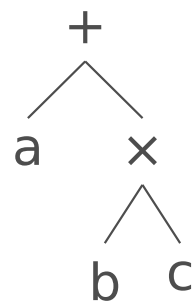
- Парсер принимает на вход строку и выдает дерево
- Принтер, наоборот, превращает дерево в строку

String $\xrightarrow{\text{parse}}$ Tree

Tree $\xrightarrow{\text{print}}$ String

a + b × c





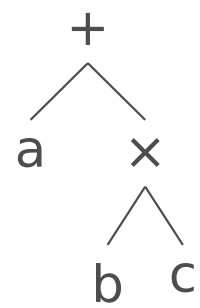
a + b × c

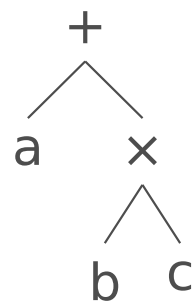
- Парсер принимает на вход строку и выдает дерево
- Принтер, наоборот, превращает дерево в строку

String $\xrightarrow{\text{parse}}$ Tree

Tree $\xrightarrow{\text{print}}$ String

a + b × c

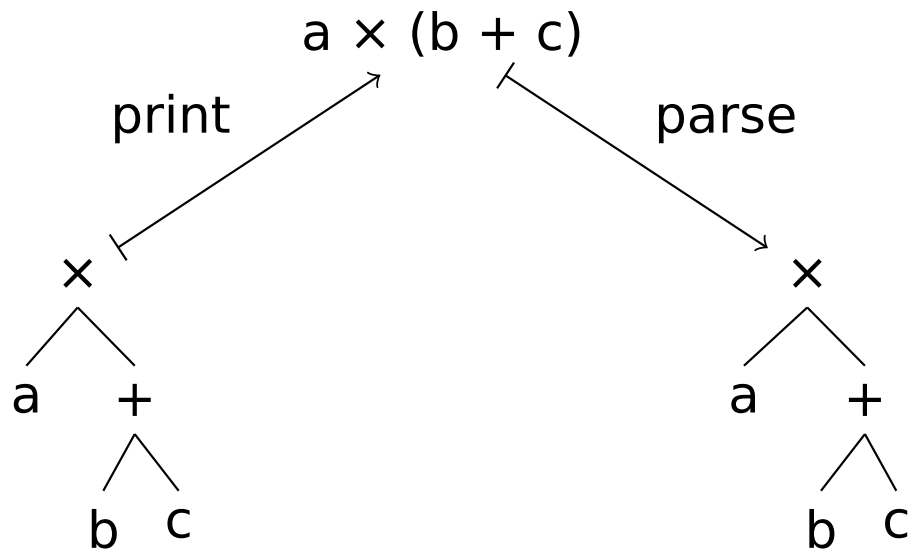




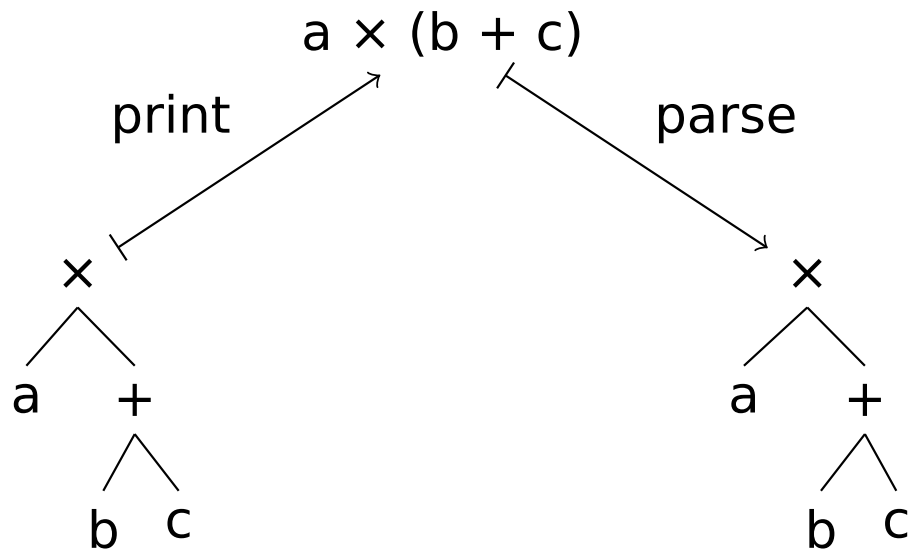
a + b × c

- Полезное свойство принтера: parse ∘ print = id

- Полезное свойство принтера: parse ∘ print = id



- Полезное свойство принтера: parse ∘ print = id

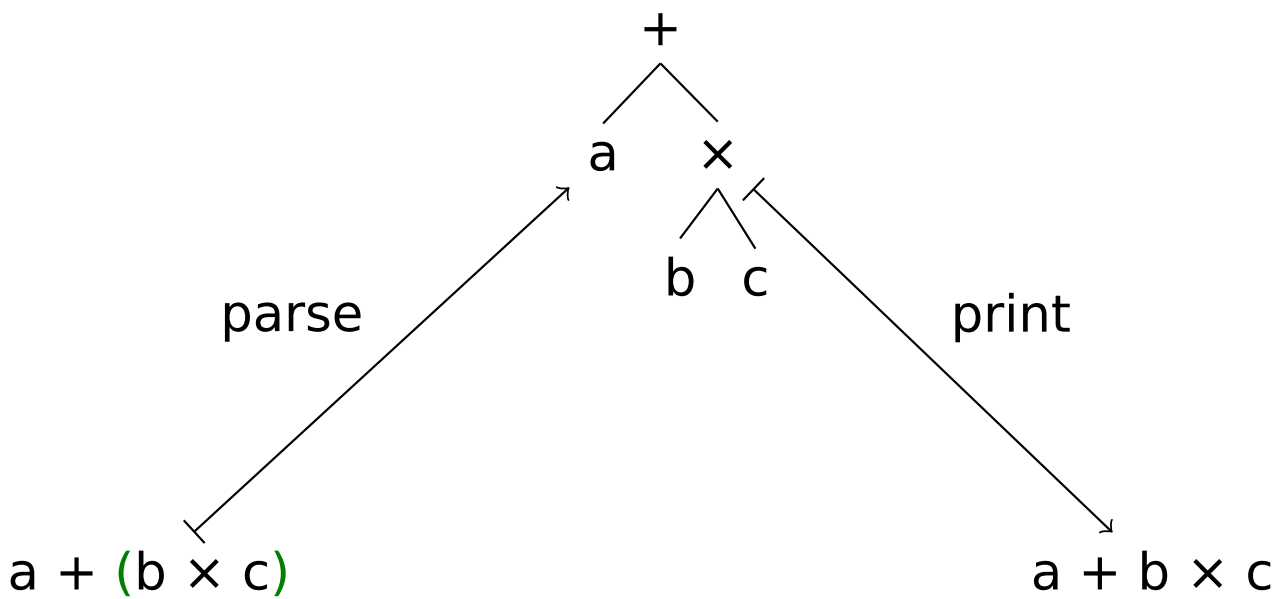


- В дереве нет скобок (всё и так однозначно)
- Принтер при печати добавляет необходимые скобки

- Полезное свойство №1: $\text{parse} \circ \text{print} = \text{id}$

- Полезное свойство №1: $\mathrm{parse} \circ \mathrm{print} = \mathrm{id}$
- Полезное свойство №2: $\mathrm{print} \circ \mathrm{parse} = \mathrm{id}$

- Полезное свойство №1: parse ∘ print = id
- Полезное свойство №2: print ∘ parse = id



- Принтеру неоткуда узнать, что там были скобки

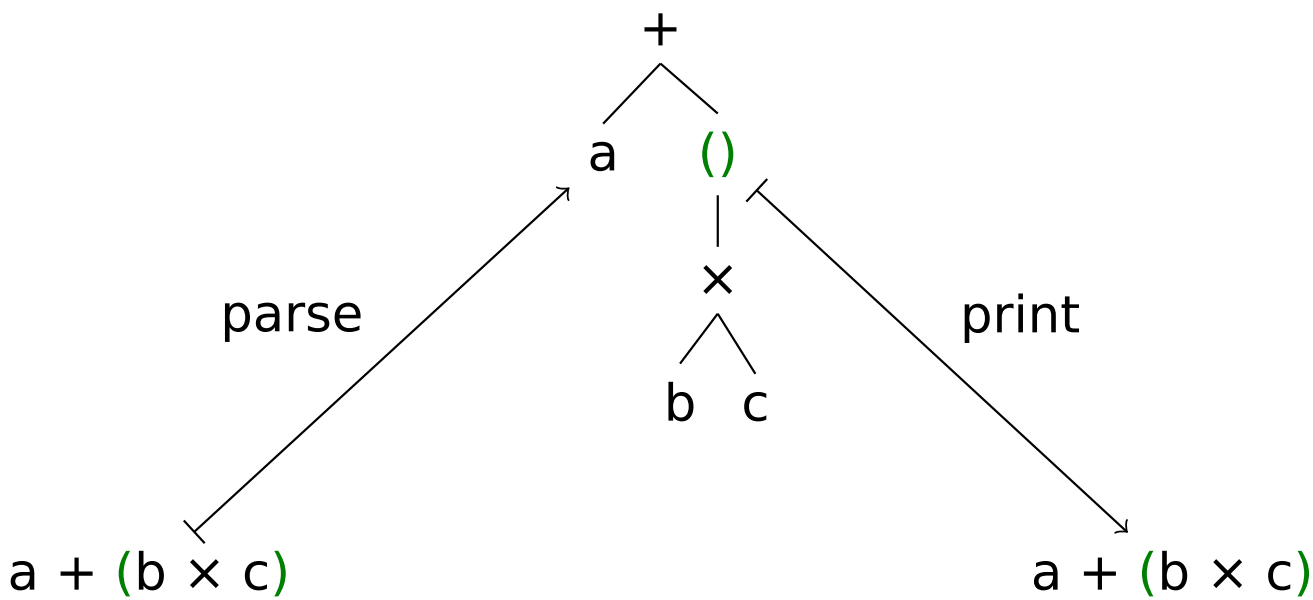
- Обязательно надо сохранять скобки в дереве

- Обязательно надо сохранять скобки в дереве

- Давайте парсер будет сохранять все скобки
- Принтер будет печатать всё как есть

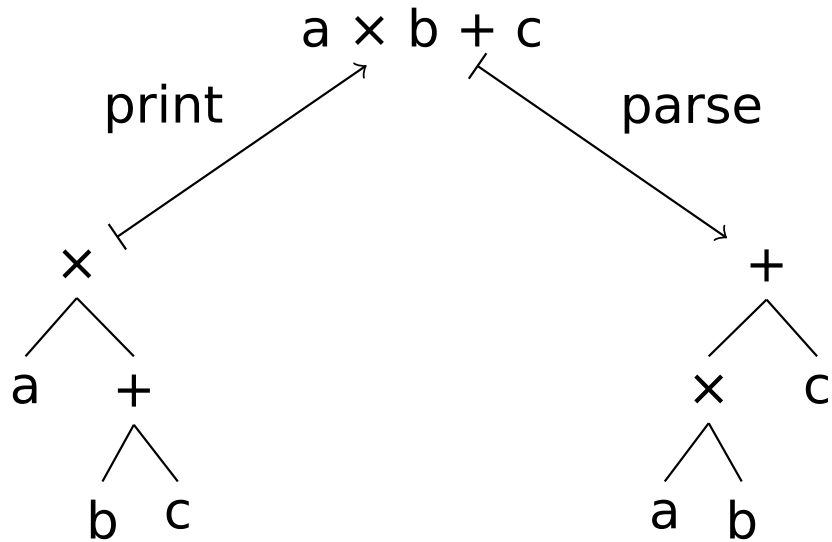
- Обязательно надо сохранять скобки в дереве

- Давайте парсер будет сохранять все скобки
- Принтер будет печатать всё как есть



- Обязательно надо сохранять скобки в дереве

- Давайте парсер будет сохранять все скобки
- Принтер будет печатать всё как есть



- Просто печатать всё как есть нельзя

- Надо и хранить скобки в дереве
- … и добавлять новые при печати

- Какие скобки надо хранить, а какие можно не хранить?
- Как печатать?

- Надо и хранить скобки в дереве
- … и добавлять новые при печати

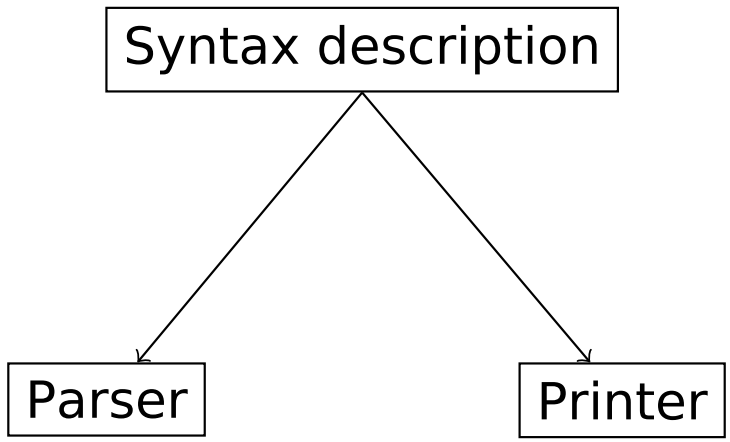
- Какие скобки надо хранить, а какие можно не хранить?
- Как печатать?

Задача

1. Ответить на эти вопросы
2. Доказать, что предложенный способ работает (на бумажке/Agda/Coq/vclang/…)

Вторая часть: описание синтаксиса


Parser
Printer


Syntax description
Parser
Printer

Invertible Syntax Descriptions: Unifying Parsing and Pretty Printing

Tillmann Rendel Klaus Ostermann
University of Marburg, Germany

Abstract

Parsers and pretty-printers for a language are often quite similar, yet both are typically implemented separately, leading to redundancy and potential inconsistency. We propose a new interface of *syntactic descriptions*, with which both parser and pretty-printer can be described as a single program. Whether a syntactic description is used as a parser or as a pretty-printer is determined by the implementation of the interface. Syntactic descriptions enable programmers to describe the connection between concrete and abstract syntax once and for all, and use these descriptions for parsing or pretty-printing as needed. We also discuss the generalization of our programming technique towards an algebra of partial isomorphisms.

Categories and Subject Descriptors D.3.4 [*Programming Techniques*]: Applicative (Functional) Programming

General Terms Design, Languages

Keywords embedded domain specific languages, invertible com-

parser DSL (Leijen and Meijer 2001), and a pretty printer EDSL (Hughes 1995). However, these EDSLs are completely independent, which precludes the use of a single embedded program to specify both parsing and pretty printing. This means that due to the dual nature of parsing and pretty-printing a separate specification of both is at least partially redundant and hence a source of potential inconsistency.

This work addresses both invertible computation and the unification of parsing and pretty printing as separate, but related challenges. We introduce the notion of *partial isomorphisms* to capture invertible computations, and on top of that, we propose a language of *syntax descriptions* to unify parsing and pretty printing EDSLs. A syntax description specifies a relation between abstract and concrete syntax, which can be interpreted as parsing a concrete string into an abstract syntax tree in one direction, and pretty printing an abstract syntax tree into a concrete string in the other direction. This dual use of syntax descriptions allows a programmer to specify the relation between abstract and concrete syntax once and for all, and use these descriptions for parsing or printing as needed.

Задача

1. Придумать способ описания синтаксиса
2. Закодить генератор парсеров и принтеров из такого описания (на любом разумном языке программирования)

Контакты

kirelagin везде:

- kirelagin@gmail.com
- telegram.me/kirelagin
- vk.com/kirelagin
- ...