Overview of our Languages

Roger Burtonpatel

November 24, 2023

This document is meant as an accompaniment to the grammars presented in `syntax-judgement-V-.pdf`*,* `syntax-judgement-P+.pdf`*, and* `decision-trees.pdf`*.*

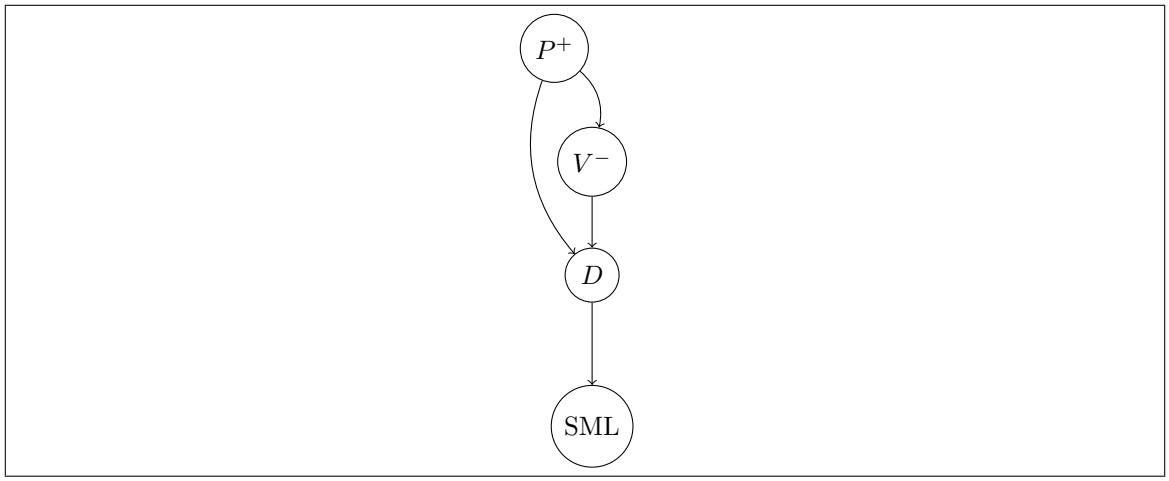

Figure 1: Flow of language translation.

1 The Three Languages

The first of our languages is P^+. This is the language of patterns. We know we can compile patterns into decision trees. Our next language is V^-, the language of equations. This uses the decision making construct *if-fi* to perform different operations based on the form of a value.

Both of these languages theoretically can be compiled to our third language, D: the language of decision trees.

2 Language Structure

The three langauges look similar: they each have value constructors and a 'decision-making construct' to deal with constructed data. In P^+, the construct is pattern-matching; in V^-, it is the guarded expression; in D, it is the decision tree.

Of note in both $V^-$and $D$is that the 'decision-making construct' is annotated with an α. This annotation gives us type flexibility on the right-hand side of the *terminating* case for each construct ($\rightarrow$ `exp` in $V^-$and the match node in D.)

Because of the α, the right-hand side becomes immaterial: we don't care about what we're returning; we care about the decision-making construct that gets us there. Whether it's a single value (ML-style) a sequence of values (Verse-style), or even something else, the α represents *any* ultimate result of "making a decision," and it's the ways in which we make decisions that we truly care about examining. By making the return result both polymorphic and abstract, we eschew the need to worry about its type and compatibility with other results of otherwise-equivalent trees.

3 Translation and Equivalency

We aim to translate P^+ to both V^- and D (Figure 1). By showing both languages can be translated to decision trees, we show both express a common property: that evaluation at runtime does not

backtrack, i.e., no individual part of a value used to make decisions (called a `scrutinee`) is examined more than once. Showing V^-expresses this no-backtracking property (and, by extension, the "no-logical-variables-at-runtime" property, is a core aim of this work.)

The second important translation is from V^- to D. Recall the α from earlier: the polymorphic abstract type in both g_α and $\mathcal{D}_\alpha$ shows an equivalence preserved by translation.

Rough- want to hack this out with you:
The equivalence is this: final, returned right-hand sides of 'decision-making constructs' can be anything.