A Formal Model and Interactive Visualization of miniKanren Search Semantics

Brysen Pfingsten Jason Hemann

Abstract

Mechanized executable semantics are a valuable tool for implementers and users alike to better understand their programs' behavior. The need is particulary acute in the context of novel logic programming languages with complex search strategies. Few such tools exist for logic-based programming languages however, and existing work only models languages' search behavior at a somewhat coarse-grained level. **In this work, we present the first small-step semantics for a logic language that models interleaving search at the level of individual goal execution.** The model is implemented in PLT Redex, a Racket-based semantics workbench, and allows users to step through the language's execution at a sub-interleave level.

We also present a visualization tool implemented through interactive JS in the browser with our miniKanren Redex model at its core. This visualizer allows users to input their own miniKanren programs as Racket source code; these programs are transpiled to the language of our model. At each step, users can easily see the evaluation context of their programs, trace goals both to and from the source code and the search tree visualization, and readily understand the history of the computation that led parts of the current state to arise. This not only enables better performance analysis and a unique, visual way of debugging but also allows novice miniKanren programmers to more readily understand the semantics and structure underlying the miniKanren search evolution.

Grammar

$$
\begin{array}{lll}
p & ::= & \texttt{(prog } \Gamma\ e\texttt{)} \\
\Gamma & ::= & \texttt{((}r_!\ d\ g\texttt{)}\ \ldots\texttt{)} \\
d & ::= & \texttt{(}x_!\ \ldots\texttt{)} \\
s & ::= & \texttt{()} \mid \texttt{(}g\ \sigma\texttt{)} \mid \texttt{(}s \rightarrow s\texttt{)} \mid \texttt{(}s \leftarrow s\texttt{)} \\
 & \mid & \texttt{((}\top\ \sigma\texttt{)} + s\texttt{)} \mid \texttt{(}s\ \times\ g\texttt{)} \\
 & \mid & \texttt{(proceed ((}r\ t\ \ldots\texttt{)}\ \sigma\texttt{)} \\
 & \mid & \texttt{(delay } s\texttt{)} \\
g & ::= & \top \mid \texttt{(}t\ \texttt{=?}\ t\texttt{)} \mid \texttt{(}r\ t\ldots\texttt{)} \\
 & \mid & \texttt{(}g \vee g\texttt{)} \mid \texttt{(}g \wedge g\texttt{)} \mid \texttt{(}\exists\ d\ g\texttt{)} \\
c & ::= & \textsf{natural} \\
x & ::= & \text{(variable-prefix x:)} \\
r & ::= & \text{(variable-prefix r:)} \\
t & ::= & c \mid \textsf{boolean} \mid \textsf{string} \mid \textsf{symbol} \\
 & \mid & x \mid \texttt{empty} \mid \texttt{(}t\ :\ t\texttt{)} \\
\sigma & ::= & \texttt{(state } sub\ c\texttt{)} \\
sub & ::= & \texttt{((}c\ t\texttt{)}\ \ldots\texttt{)}
\end{array}
$$

Interleaving

At the core of miniKanren search is the idea of the interleave. In order to ensure a more fair search, each relation call is delayed until it reaches the top of our evaluation context. Traditionally, this operation is done by flipping the tree such that the newly delayed relation call must wait for the other side of the tree to complete its computation or encounter another relation call to be invoked. Visually, however, this transformation is confusing as it requires the programmer to recalibrate themselves with the new shape of the tree. As such, we implement a *railway model* with directed tree disjunctions that preserves the shape of the tree while allowing for interleave behavior.

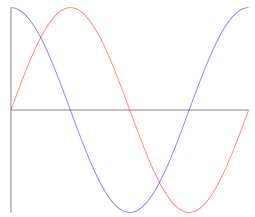

Figure 1. Another figure caption.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam vel dapibus erat. Morbi quis leo congue, lobortis augue bibendum, malesuada neque. Duis ullamcorper quis orci sed consequat. Nam pellentesque ullamcorper tempor. Duis eget nulla blandit, vulputate orci vitae, ullamcorper ligula. Mauris a urna ac massa dignissim scelerisque sed et augue. Donec eget urna vitae neque elementum pellentesque et eget enim. Praesent a fermentum nibh. Nullam eu nibh neque.

A highlighted block containing some math

A different kind of highlighted block.

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$$

Interdum et malesuada fames $\{1, 4, 9, \ldots\}$ ac ante ipsum primis in faucibus. Cras eleifend dolor eu nulla suscipit suscipit. Sed lobortis non felis id vulputate.

A heading inside a block

Praesent consectetur mi $x^2 + y^2$ metus, nec vestibulum justo viverra nec. Proin eget nulla pretium, egestas magna aliquam, mollis neque. Vivamus dictum $\mathbf{u}^\intercal\mathbf{v}$ sagittis odio, vel porta erat congue sed. Maecenas ut dolor quis arcu auctor porttitor.

Another heading inside a block

Sed augue erat, scelerisque a purus ultricies, placerat porttitor neque. Donec $P(y \mid x)$ fermentum consectetur $\nabla_x P(y \mid x)$ sapien sagittis egestas. Duis eget leo euismod nunc viverra imperdiet nec id justo.

Nullam vel erat at velit convallis laoreet

Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Phasellus libero enim, gravida sed erat sit amet, scelerisque congue diam. Fusce dapibus dui ut augue pulvinar iaculis.

First column	Second column	Third column	Fourth
Foo	13.37	384,394	α
Bar	2.17	1,392	β
Baz	3.14	83,742	δ
Qux	7.59	974	γ

Table 1. A table caption.

Donec quis posuere ligula. Nunc feugiat elit a mi malesuada consequat. Sed imperdiet augue ac nibh aliquet tristique. Aenean eu tortor vulputate, eleifend lorem in, dictum urna. Proin auctor ante in augue tincidunt tempor. Proin pellentesque vulputate odio, ac gravida nulla posuere efficitur. Aenean at velit vel dolor blandit molestie. Mauris laoreet commodo quam, non luctus nibh ullamcorper in. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.