Stack Graphs

Single Scope Stack (current)

Notation

$$
\begin{array}{rrcl}
\textit{symbol} & x & & \\
\textit{scope} & s & & \\
\textit{node} & N & := & s \mid \downarrow x \mid \downarrow x/s \mid \uparrow x \mid \uparrow x/\bullet \mid \mathsf{JUMP} \mid \mathsf{DROP} \\
\textit{edge} & e & := & N \to N \\
\textit{scoped symbol} & a & := & x \mid x/\Sigma \\
\textit{symbol stack} & \mathrm{X} & := & () \mid \mathrm{X} \cdot a \\
\textit{scope stack} & \Sigma & := & \diamondsuit \mid \Sigma \cdot s \\
\textit{path} & p & := & (\vec{N}, \mathrm{X}, \Sigma)
\end{array}
$$

Valid Paths

$$\textsc{Scope}\frac{(\vec{N} \cdot N_1, \mathrm{X}, \Sigma) \quad N_1 \to s}{(\vec{N} \cdot N_1 \cdot s, \mathrm{X}, \Sigma)}$$

$$\textsc{Push}\frac{(\vec{N} \cdot N_1, \mathrm{X}, \Sigma) \quad N_1 \to \downarrow x}{(\vec{N} \cdot N_1 \cdot \downarrow x, \mathrm{X} \cdot x, \Sigma)} \qquad \textsc{Pop}\frac{(\vec{N} \cdot N_1, \mathrm{X} \cdot x, \Sigma) \quad N_1 \to \uparrow x}{(\vec{N} \cdot N_1 \cdot \uparrow x, \mathrm{X}, \Sigma)}$$

$$\textsc{PushScoped}\frac{(\vec{N} \cdot N_1, \mathrm{X}, \Sigma) \quad N_1 \to \downarrow x/s}{(\vec{N} \cdot N_1 \cdot \downarrow x/s, \mathrm{X} \cdot x/(\Sigma \cdot s), \Sigma)} \qquad \textsc{PopScoped}\frac{(\vec{N} \cdot N_1, \mathrm{X} \cdot x/\Sigma', \Sigma) \quad N_1 \to \uparrow x/\bullet}{(\vec{N} \cdot N_1 \cdot \uparrow x/\bullet, \mathrm{X}, \Sigma')}$$

$$\textsc{Jump}\frac{(\vec{N} \cdot N_1, \mathrm{X}, \Sigma \cdot s) \quad N_1 \to \mathsf{JUMP}}{(\vec{N} \cdot N_1 \cdot \mathsf{JUMP} \cdot s, \mathrm{X}, \Sigma)} \qquad \textsc{Drop}\frac{(\vec{N} \cdot N_1, \mathrm{X}, \Sigma) \quad N_1 \to \mathsf{DROP}}{(\vec{N} \cdot N_1 \cdot \mathsf{DROP}, \mathrm{X}, \diamondsuit)}$$

Scope Context (proposed)

Instead of a single scope stack Σ in a path, we have a scope context Ψ, consisting of a stack of symbol stacks. This context holds all active pops at this point. The elements in a scope stack Σ are scopes with an attached context Ψ, which are the enclosing pops at the point of the push. A DROP drops the top of the scope stack stack, making the directly enclosing pop active again. A JUMP jumps to the top element of the top scope stack in the context, drops the enclosing context, and restores the enclosing context of the scope that was jumped to.

The scope context represents lexical nesting, while the scope stack represents a call stack. A JUMP takes us to the caller, i.e., the previous element in the current scope stack. At this point the current context is irrelevant, and the context of the caller is restored. A DROP takes us to the enclosing scope. At that point, the current call stack is irrelevant, and the call stack of the enclosing scope is restored.

With this semantics, DROP behaves as currently, if the scope stack stack Ψ is a singleton, but correctly restores any earlier pops that were there. Therefore, I think it does not require changes to the existing rules.

Notation

$$
\begin{array}{rrcl}
\textit{symbol stack} & \mathrm{X} & := & () \mid \mathrm{X} \cdot a \\
\textit{scope stack} & \Sigma & := & \diamondsuit \mid \Sigma \cdot s[\Psi] \\
\textit{scope context} & \Psi & := & \blacklozenge \mid \Psi \cdot \Sigma \\
\textit{path} & p & := & (\vec{N}, \mathrm{X}, \Psi)
\end{array}
$$

Valid Paths

$$\text{PushScoped'}\frac{(\vec{N}\cdot N_1, \mathrm{X}, \Psi\cdot\Sigma) \quad N_1 \to \downarrow x/s}{(\vec{N}\cdot N_1\cdot\downarrow x/s, \mathrm{X}\cdot x/(\Sigma\cdot s[\Psi]), \Psi\cdot\Sigma)} \qquad \text{PopScoped'}\frac{(\vec{N}\cdot N_1, \mathrm{X}\cdot x/\Sigma, \Psi) \quad N_1 \to \uparrow x/\bullet}{(\vec{N}\cdot N_1\cdot\uparrow x/\bullet, \mathrm{X}, \Psi\cdot\Sigma)}$$

$$\text{Jump'}\frac{(\vec{N}\cdot N_1, \mathrm{X}, \Psi\cdot(\Sigma\cdot s[\Psi'])) \quad N_1 \to \mathsf{JUMP}}{(\vec{N}\cdot N_1\cdot\mathsf{JUMP}\cdot s, \mathrm{X}, \Psi'\cdot\Sigma)} \qquad \text{Drop'}\frac{(\vec{N}\cdot N_1, \mathrm{X}, \Psi\cdot\Sigma) \quad N_1 \to \mathsf{DROP}}{(\vec{N}\cdot N_1\cdot\mathsf{DROP}, \mathrm{X}, \Psi)}$$

Additionally, we should consider a rule that allows dropping empty scope contexts (keeping them empty). This rule ensures that a reference inside a nested scope can resolve to the definition in the surrounding context, even when it is not part of an application. The alternative, having a path with a drop node, and one without, to the surrounding context, can cause wrong results, because even when there is context, the drop could be ignored.

$$\text{DropEmpty'}\frac{(\vec{N}\cdot N_1, \mathrm{X}, \blacklozenge) \quad N_1 \to \mathsf{DROP}}{(\vec{N}\cdot N_1\cdot\mathsf{DROP}, \mathrm{X}, \blacklozenge)}$$

Plan

Introducing these changes would require the following to be done:

- Adapt all parts of the formalism: partial paths, partial path composition.
- Adapt algorithm descriptions (pseudocode).
- Adapt Rust implementation: querying and indexing.
- Adapt Go service to use new Rust implementation for querying by lifting existing data to the new format. Keep Go indexing implementation for the time being.
- Start using Rust indexing implementation to produces data in new format. The data lifting stays in place until all data is updated.