motivation

Today, the onus of coordinating SDN software falls on the admin to write modular programs, with:
- Modularization prefixed in specific programming constructs varies from one DSL to another.
- Manual (control flow) composition relies on the internalized knowledge of experienced admins.
Making modularization an architectural primitive of systems organized around data flow:
- Data-flow modularization architecture in pre-SDN era.
- SDN building blocks (modules) are extremely flexible and keep evolve — conflicts among modules?
- Interact in arbitrary ways — not terminate?

Towards a modularized data-flow architecture:
- Software as semantic units that maintain properties.
- Consistency / termination by semantic analysis.

background: residue method

Residue represents the integrity constraint's effect on the network data:
- Integrity constraint as the subsuming clause.
- Network state (negated relation) as the subsumed clause.
- The fragment at the bottom of refutation tree.

Residues for each module (p,d):
- The effect of module i on d.
- Every d_k is associated with a unique residue (head, body).
- A single operation op _k.

Consistency and termination analyzer

Consistency:
- Modules agree on one single unique operation over the network (data).
- Data-flow execution (no control flow spec — master program — is needed).

Example: Consistency analysis with residues

fw_1 module — firewall applied in slice 1:
fw_1 \leftarrow flow(\text{F}, \text{Y}), block(\text{X}, \text{Y}) where
\begin{align*}
\text{flow}(\text{F}, \text{Y}) &\leftarrow flow(\text{F}, \text{Y}), \text{slice}_1(\text{X}, \text{Y}) \\
\text{flow}(\text{F}, \text{Y}) &\leftarrow block(\text{F}, \text{Y}), slice_1(\text{X}, \text{Y})
\end{align*}

s_2 module — manage flow of slice 2:
s_2 \leftarrow flow(\text{F}, \text{Y}) \leftarrow flow(\text{F}, \text{Y}), \text{slice}_2(\text{X}, \text{Y})

rfw_{1,1}:
flow(\text{F}, \text{X}, \text{Y}) \leftarrow block(\text{F}, \text{X}, \text{Y})
\begin{array}{c}
\text{flow}(\text{F}, \text{X}, \text{Y}) \\
\text{flow}(\text{F}, \text{X}, \text{Y})
\end{array}

rfw_{1,2}:
flow(\text{F}, \text{X}, \text{Y}) \leftarrow block(\text{F}, \text{X}, \text{Y}), slice_1(\text{X}, \text{Y}), slice_2(\text{X}, \text{Y})

Firewall in isolated slices will not affect one another.