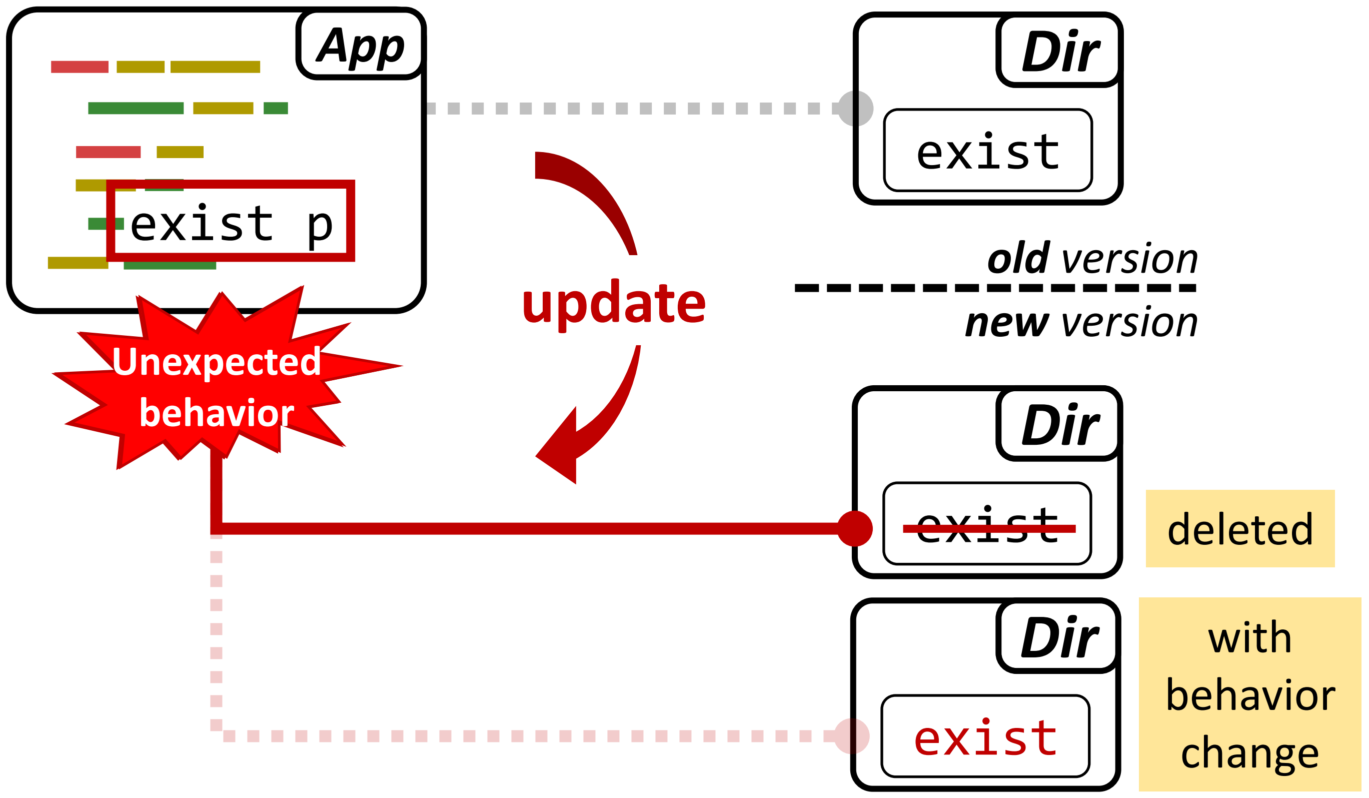
A Functional Programming Language with Versions

Yudai Tanabe[a)] Luthfan Anshar Lubis[a)]
Tomoyuki Aotani[b)] Hidehiko Masuhara[a)]

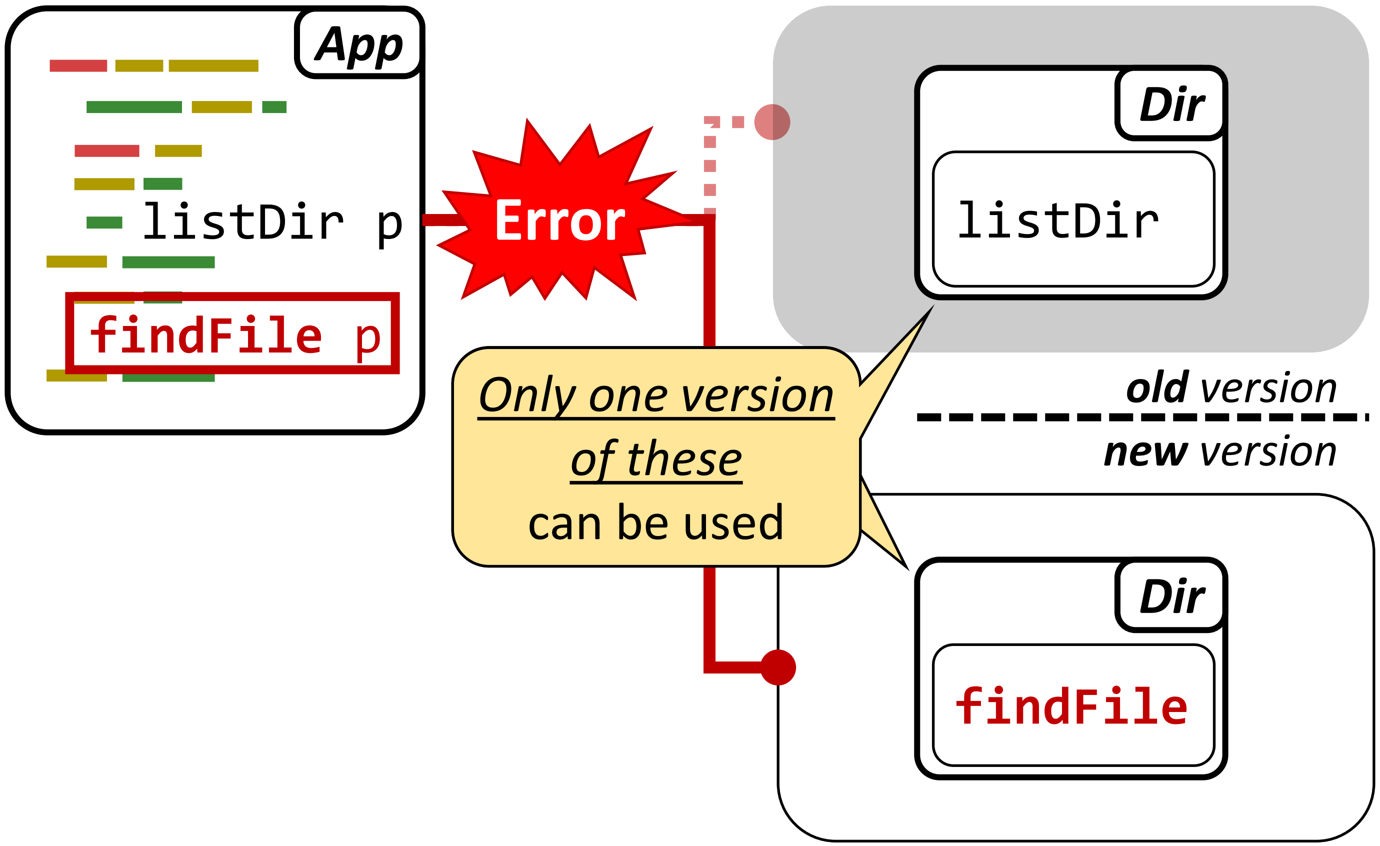
(a) Tokyo Institute of Technology (b) Mamezou

Accepted by ‹Programming› 2022

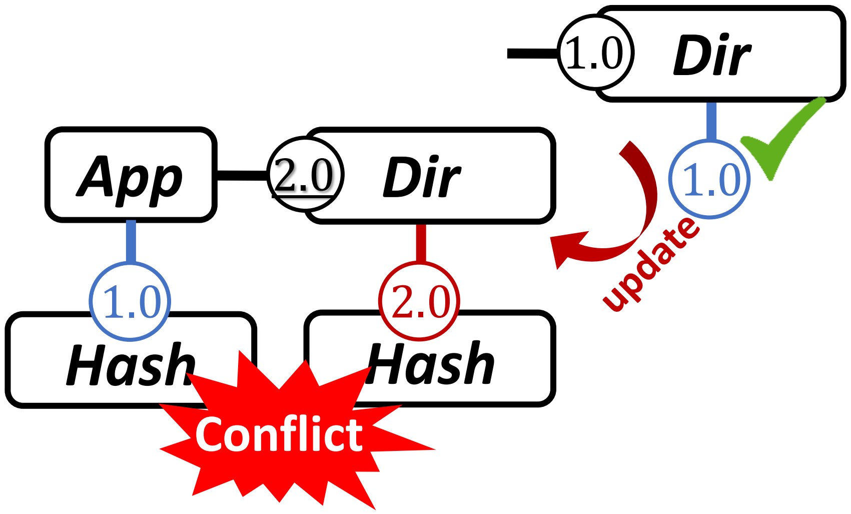
Dependency Update May Break Software



Limitation of Existing Languages: *One-version-at-a-time*



Indirect Dependencies Complicate the Problem (Dependency Hell)



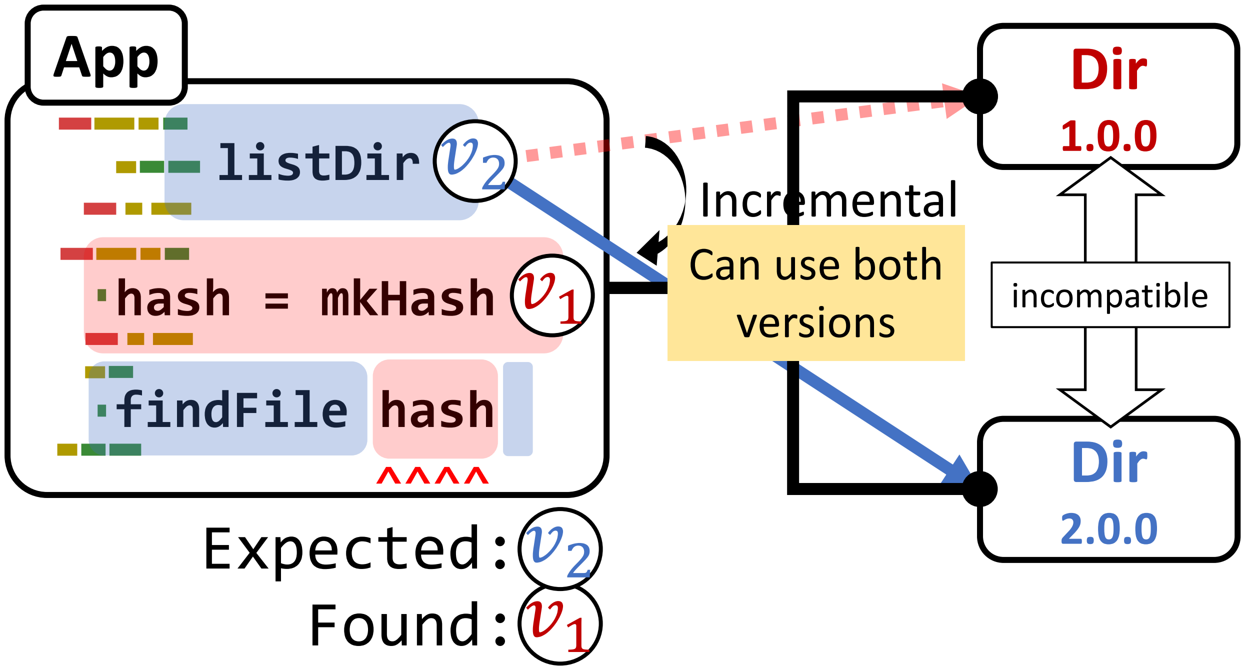
Can detect conflicts, but not resolve

- Increasing update costs
  - Lead to version locking[Preston-Werner'13]
  - Discourage users from updates[Bavota'15]

Programming Language with Versions

Handle multiple versions
in one client

Detect conflicting usage
within a program

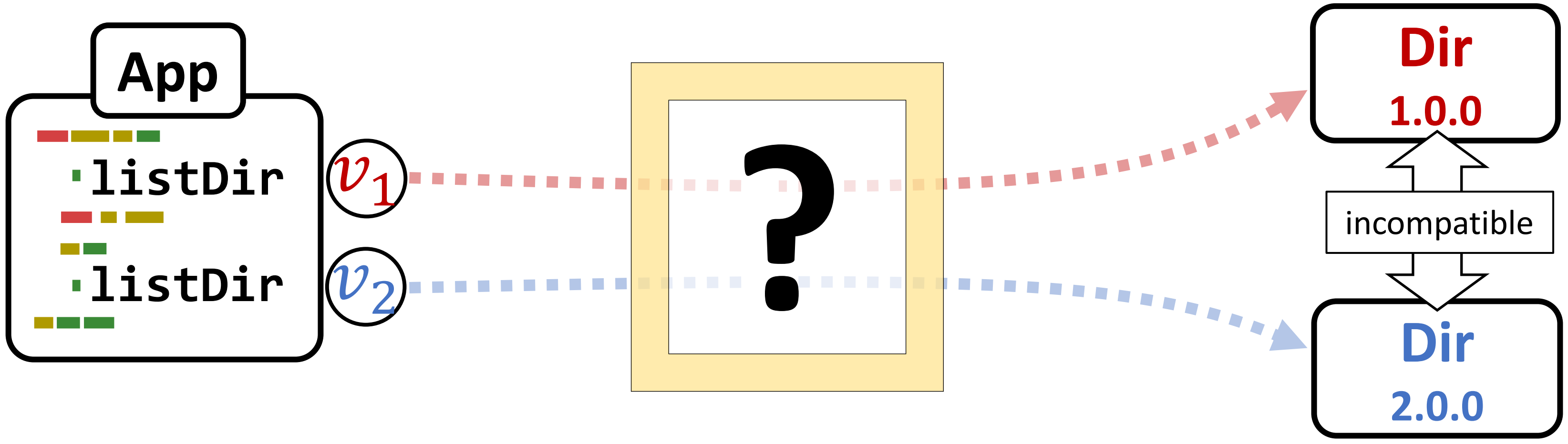


Proposal

What Language Features Do We Need?

Handle multiple versions
in one client

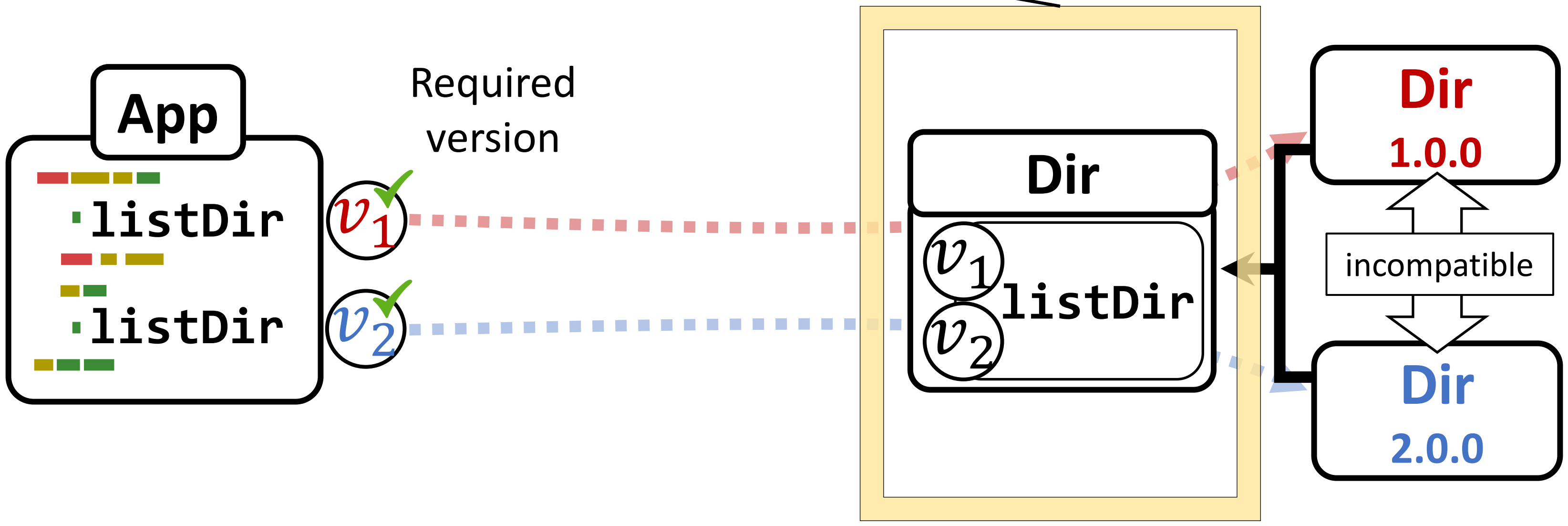
Detect conflicting usage
within a program



Proposal

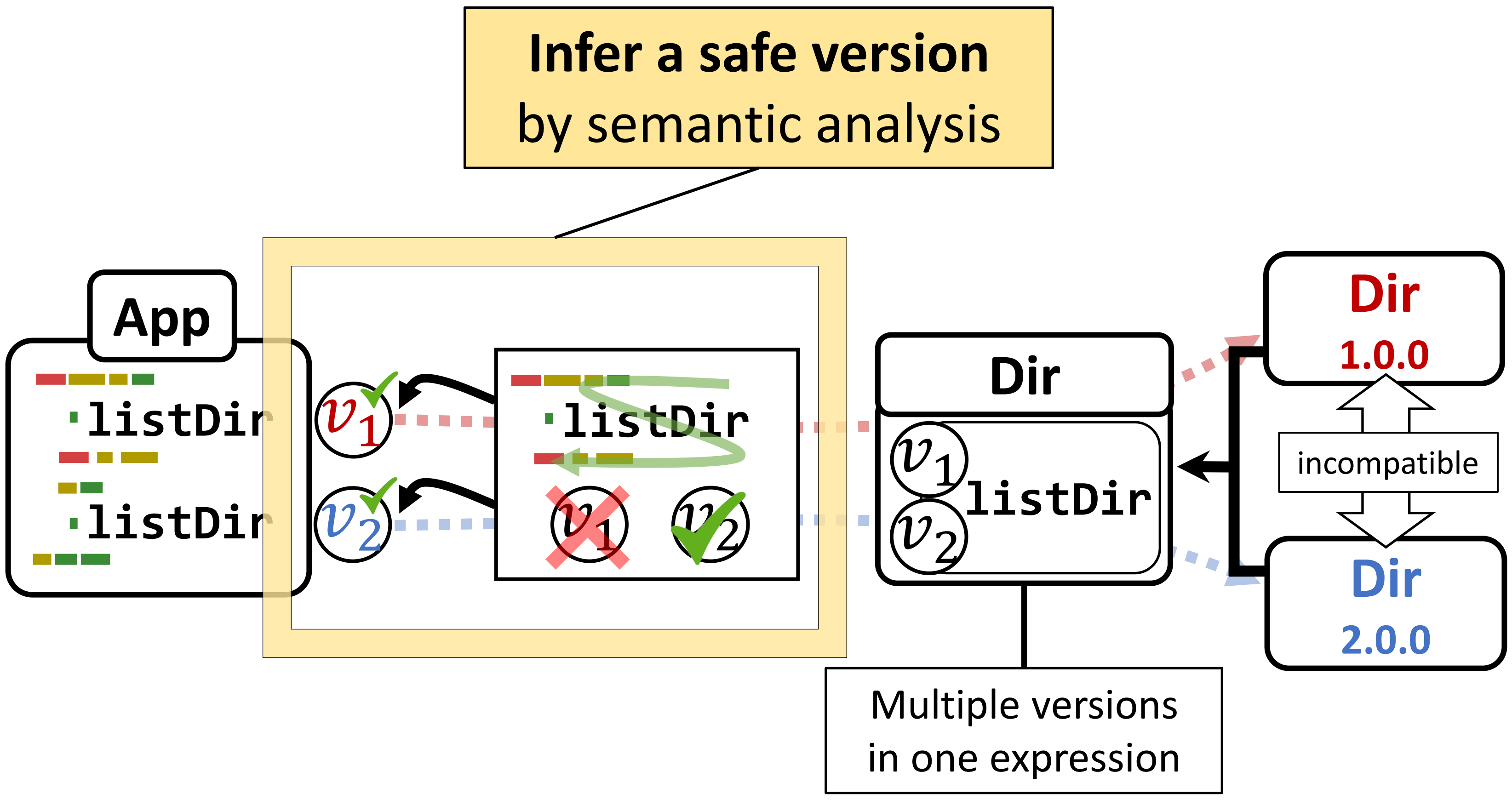
Multiple Versions in One Expression

One expression can provide different values with respect to a required version



Proposal

Version Inference

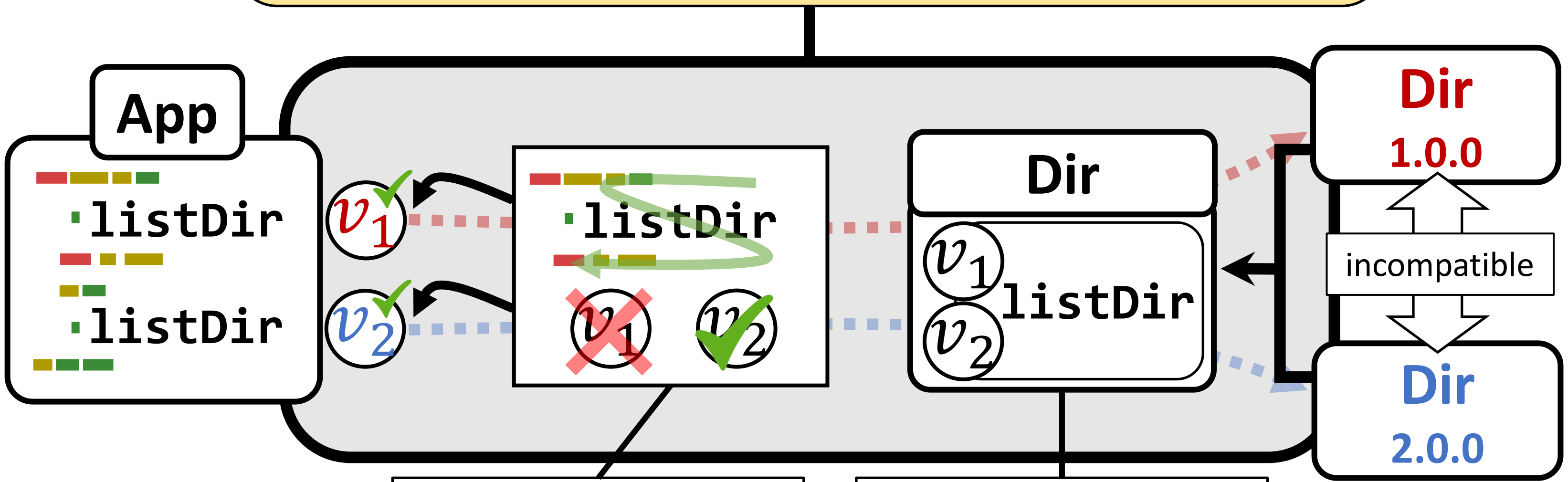


Proposal

Language Overview

Core calculus $\lambda_{\mathbf{VL}}$
[COP'18, ‹Programming›'22]

- ***Versioned values*** to hold multiple versions of definitions
- ***Type system*** to analyze available versions



Semantic analysis to specify version

Multiple versions in one expression

Core Calculus

Versioned Values

- **Versioned records** $\{ \overline{l_i = t_i} \}$

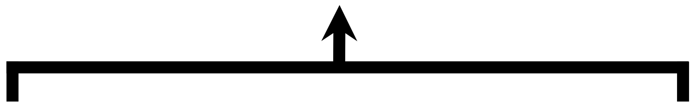
  Multiple versions in one value

Versioned labels

$$l ::= [\overline{\underbrace{M_i}_{\textit{Module name}} \mapsto \underbrace{V_i}_{\textit{Version number}}}]$$

$$findFile = \left\{ \begin{array}{l} l_1 = \backslash\text{h. listFiles} \ldots \\ l_2 = \backslash\text{h. lookup} \ldots \end{array} \right\}$$

$$l_1 = [\text{Dir} \mapsto 1.0.0]$$
$$l_2 = [\text{Dir} \mapsto 2.0.0]$$



```
\h -> …
  listFiles $
  filter …
```

Module Dir
Version 1.0.0

```
\h -> …
  lookup table h $
  getFileTable …
```

Module Dir
Version 2.0.0

Core Calculus

Versionwise Function Application

$$\boxed{\begin{array}{l}\text{let } [f] = \mathit{findFile} \text{ in} \\ \text{let } [x] = \mathit{hash} \text{ in} \\ [f\ x]\end{array}} \approx \left\{\begin{array}{l} l_1 = (\backslash\text{h. listFiles} \ldots)(acbd \ldots) \\ l_2 = (\backslash\text{h. lookup} \ldots)\ (8cdc \ldots) \\ \quad - \end{array}\right\}$$

$$\begin{array}{cc} \mathit{findFile} & \mathit{hash} \\ \left\{\begin{array}{l} l_1 = \backslash\text{h. listFiles} \ldots \\ l_2 = \backslash\text{h. lookup} \ldots \\ \quad - \end{array}\right\} & \left\{\begin{array}{l} l_1 = acbd \ldots \\ l_2 = 8cdc \ldots \\ l_3 = 73fe \ldots \end{array}\right\} \end{array}$$

Core Calculus

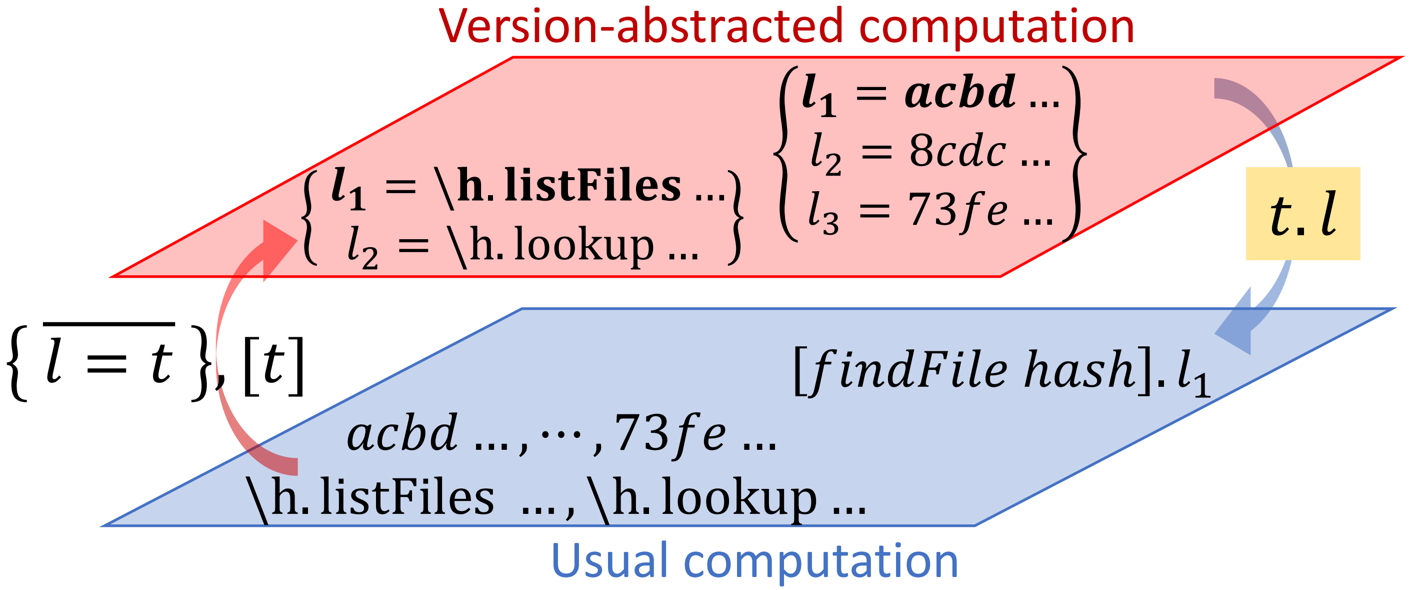
Dynamic Semantics

Extractions $t.l$: Evaluate t in a specific version l

$$\boxed{\begin{array}{l}\text{let } [f] = \mathit{findFile} \text{ in} \\ \text{let } [x] = \mathit{hash} \text{ in} \\ [f\ x].l_1\end{array}}$$

$\longrightarrow^{*}$ $(\backslash\mathbf{h.listFiles} \ldots)(\boldsymbol{acbd} \ldots)$

$\longrightarrow$ `/home/usr/proj/foo`


Version-abstracted computation
$\left\{\begin{array}{l}\boldsymbol{l_1} = \backslash\mathbf{h.listFiles} \ldots \\ l_2 = \backslash\mathrm{h.lookup} \ldots\end{array}\right\}$
$\left\{\begin{array}{l}\boldsymbol{l_1 = acbd} \ldots \\ l_2 = 8cdc \ldots \\ l_3 = 73fe \ldots\end{array}\right\}$
$t.l$
$\{\overline{l = t}\}, [t]$
$[\mathit{findFile\ hash}].l_1$
$acbd \ldots, \cdots, 73fe \ldots$
$\backslash\mathrm{h.listFiles} \ldots, \backslash\mathrm{h.lookup} \ldots$
Usual computation

Core Calculus

Version Safety = Label Consistency

Inconsistency to be detected: ***No extractable version labels***

$$\begin{array}{l}\text{let } [f] = \mathit{findFile} \text{ in} \\ \text{let } [x] = \mathit{hash} \text{ in} \\ [f\ x].\,l_1\end{array} \quad ✓ \quad \approx \begin{Bmatrix} l_1 = \backslash\mathrm{h.\,listFiles}\ \ldots \\ l_2 = \backslash\mathrm{h.\,lookup}\ \ldots \end{Bmatrix} \begin{Bmatrix} l_1 = acbd\ \ldots \\ l_2 = 8cdc\ \ldots \\ l_3 = 73fe\ \ldots \end{Bmatrix}$$

$$\begin{array}{l}\text{let } [f] = \mathit{findFile} \text{ in} \\ \text{let } [x] = \mathit{hash} \text{ in} \\ [f\ x].\,l_3\end{array} \quad ✗ \quad \approx \begin{Bmatrix} l_1 = \backslash\mathrm{h.\,listFiles}\ \ldots \\ l_2 = \backslash\mathrm{h.\,lookup}\ \ldots \end{Bmatrix} \begin{Bmatrix} l_1 = acbd\ \ldots \\ l_2 = 8cdc\ \ldots \\ l_3 = 73fe\ \ldots \end{Bmatrix}$$

Core Calculus

Types of Versioned Values

Versions as Resources

Types tagged with ***a set of available version labels***

$findFile : \Box_{\underline{\{l_1,l_2\}}}(\text{Hash} \to A)$ $\quad$ $hash : \Box_{\underline{\{l_1,l_2,l_3\}}}\text{Hash}$

$$findFile = \begin{Bmatrix} l_1 = \backslash\text{h. listFiles} \dots \\ l_2 = \backslash\text{h. lookup} \dots \end{Bmatrix} \qquad hash = \begin{Bmatrix} l_1 = acbd \dots \\ l_2 = 8cdc \dots \\ l_3 = 73fe \dots \end{Bmatrix}$$

$$\begin{array}{l} \text{let } [f] = findFile \text{ in} \\ \text{let } [x] = hash \text{ in} \qquad : \Box_{\underline{\{l_1,l_2\}}} A \\ [f\ x] \end{array}$$

$$\{l_1,l_2\} \;\|\; \{l_1,l_2\} \cap \{l_1,l_2,l_3\}$$

Types of Extractions

Inspect version consistency in subterms

$$findFile : \Box_{\{l_1,l_2\}}(\text{Hash} \to A)$$
$$hash : \Box_{\{l_1,l_2,l_3\}}\text{Hash}$$

[ERROR]
Expected l_3,
but got l_1, l_2

$$\begin{array}{l} \text{let } [f] = findFIle \text{ in} \\ \text{let } [x] = hash \text{ in} \\ [f\ x].\, l_1 \end{array} \quad ✓ : A$$

$$\begin{array}{l} \text{let } [f] = \underline{findFile} \text{ in} \\ \text{let } [x] = hash \text{ in} \\ [f\ x].\, l_3 \end{array} \quad : ✗$$

because $l_1 \in \{l_1, l_2\} \cap \{l_1, l_2, l_3\}$ ✓

because $l_3 \notin \{l_1, l_2\} \cap \{l_1, l_2, l_3\}$ ✗

Type System

Coeffect Calculus

Coeffect calculus: $\ell\mathcal{R}$PCF[Brunel'14], GrMini[Orchard'19]

$t ::= \ldots \mid x \mid t_1 t_2 \mid \lambda x.t \mid [t] \mid \text{let } [x] = t_1 \text{ in } t_2$

$A ::= \ldots \mid A \to A \mid \Box_r A$

$\Gamma ::= \emptyset \mid \Gamma, x : A \mid \Gamma, x : [A]_r$

$r \in (\mathcal{R}, \oplus, 0, \otimes, 1)$

$\mathcal{R}$-parameterized type systems

Security level[Orchard'19]

$\mathcal{R} = \{\text{Irrelevant}, \text{Private}, \text{Public}\}$

Exact usage[Petricek'14]

$\mathcal{R} = \mathbb{N}$

Type System

Version Resource Semiring

Coeffect calculus: $\ell\mathcal{R}$PCF[Brunel'14], GrMini[Orchard'19]

$t ::= \ \ldots \mid x \mid t_1 t_2 \mid \lambda x.t \mid$

$[t] \mid \text{let } [x] = t_1 \text{ in } t_2$

$A ::= \ \ldots \mid A \to A \mid \Box_r A$

$\Gamma ::= \emptyset \mid \Gamma, x : A \mid \Gamma, x : [A]_r$

$r \in (\mathcal{R}, \oplus, 0, \otimes, 1)$

$\mathcal{R}$-parameterized type systems

$t ::= \ \ldots \mid \{\overline{l = t}\} \mid t.l$

versioned values con-/de-structors

Version labels

$\mathcal{R} = \{\bot, \{\overline{l_i}\}\}$

λ_{VL}

… and some corresponding typing rules

Version Awareness of λ_{VL} Type System

Additive part: *resource splitting*

$$\frac{\underline{\Gamma_1} \vdash t_1 : A \to B \qquad \underline{\Gamma_2} \vdash t_2 : A}{\underline{\Gamma_1 + \Gamma_2} \vdash t_1\ t_2 : B}\ \text{app}$$

Splitting resources for sub judgments

$$\boxed{\begin{aligned}&(\Gamma, x{:}\,[A]_r) + (\Gamma', x{:}\,[A]_s)\\ &= (\Gamma + \Gamma'), x{:}\,[A]_{r \oplus s}\end{aligned}}$$

Multiplication part: *resource demanding*

$$\frac{[\Gamma] \vdash t : A}{\underline{r * [\Gamma]} \vdash [t] : \underline{\Box_r A}}\ \text{pr}$$

$$\begin{tikzcd} r * [\Gamma] & \Box_r A \arrow[l, bend left] \end{tikzcd}$$

Requiring resources from a context

$$\boxed{r * (\Gamma, x{:}\,[A]_s) = (r \cdot \Gamma), x{:}\,[A]_{r \otimes s}}$$

"$[t]$ available in r requires
all assumptions to be available in r."

Properties

Type-safe extractions

Proved

$$[\Gamma] \vdash v : \Box_{\boldsymbol{r}} A \implies \forall \boldsymbol{l_k} \in \boldsymbol{r}. \exists t'. \begin{cases} v.\boldsymbol{l_k} \longrightarrow t' \\ [\Gamma] \vdash t' : A \end{cases}$$

Type soundness

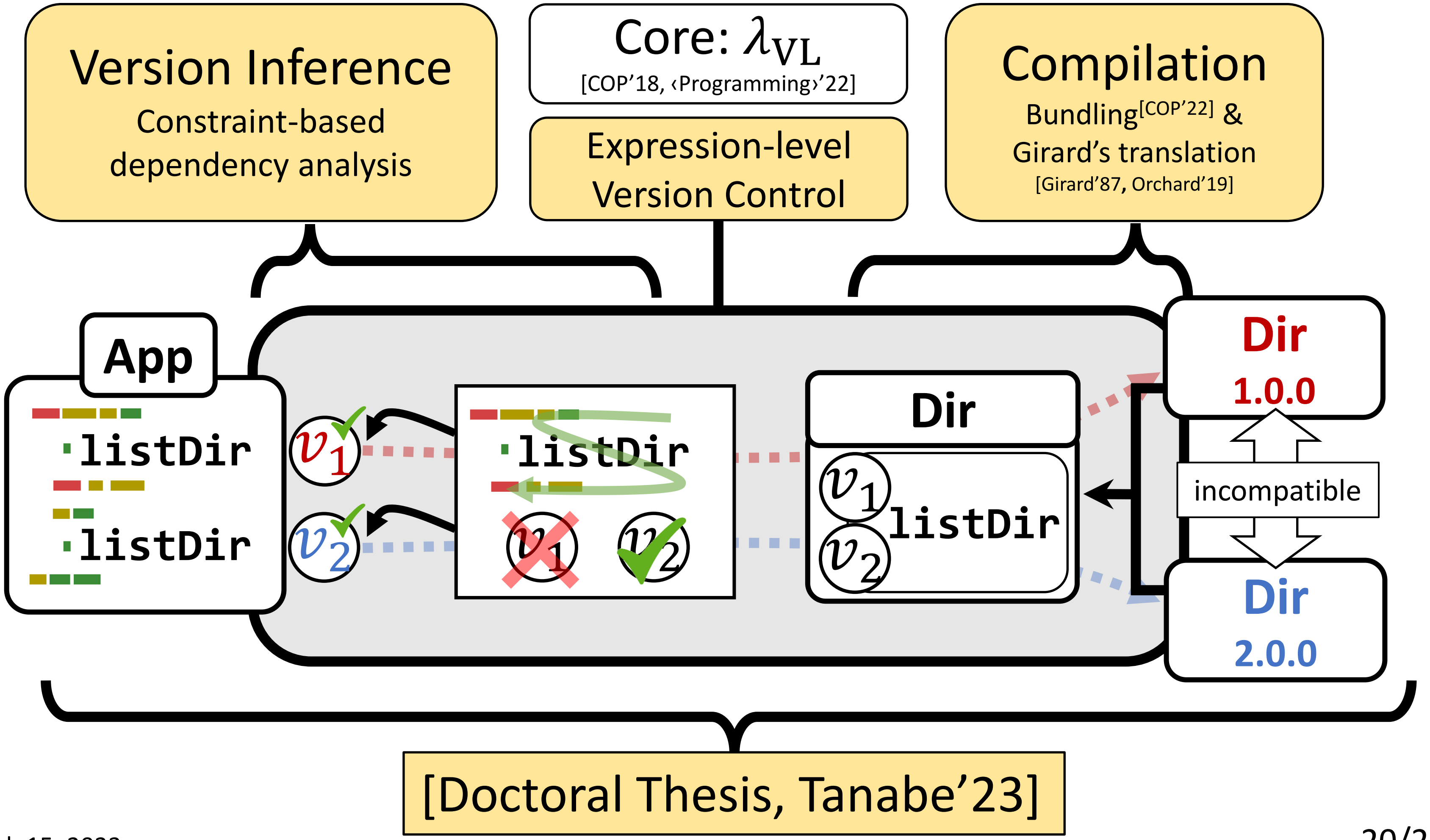
Proved

$$\Gamma \vdash t : A \land t \longrightarrow t' \implies \Gamma \vdash t' : A \qquad \text{(preservation)}$$

$$\emptyset \vdash t : A \implies \text{value}\ t \lor \exists t'. t \longrightarrow t' \qquad \text{(progress)}$$

Work in Progress

Implementation (Work After ‹Programming›’22)


Version Inference
Constraint-based dependency analysis
Core: λ_{VL}
[COP’18, ‹Programming›’22]
Expression-level Version Control
Compilation
Bundling[COP’22] & Girard’s translation
[Girard’87, Orchard’19]
App
listDir
listDir
v_1
v_2
listDir
v_1
v_2
Dir
v_1
listDir
v_2
Dir 1.0.0
incompatible
Dir 2.0.0
[Doctoral Thesis, Tanabe’23]

Future Work

Toward Support for Type Incompatibilities

Motivation: ***No support for type incompatibilities*** in λ_{VL}

$$\frac{\ldots \vdash t_1 : \textcircled{\boldsymbol{A}} \qquad \ldots \vdash t_2 : \textcircled{\boldsymbol{A}}}{\ldots \vdash \{l_1 = t_1, l_2 = t_2\} : \Box_{\{l_1,l_2\}}\boldsymbol{A}}$$

Not assuming different types of terms in different versions

$$[t] : \Box_{\{l_1,l_2\}}\boldsymbol{A}$$

Idea: ***Integrating version checking into record calculus***[Ohori'95]

$\boxed{\lambda_{\mathrm{VL}}}$ $\textcircled{?}$ $\boxed{\lambda^{let,\cdot}}$

$$x : \Box_{\{l_1\}}A \;\simeq\; x : \forall t :: \{l_1 : A\}.\, t$$

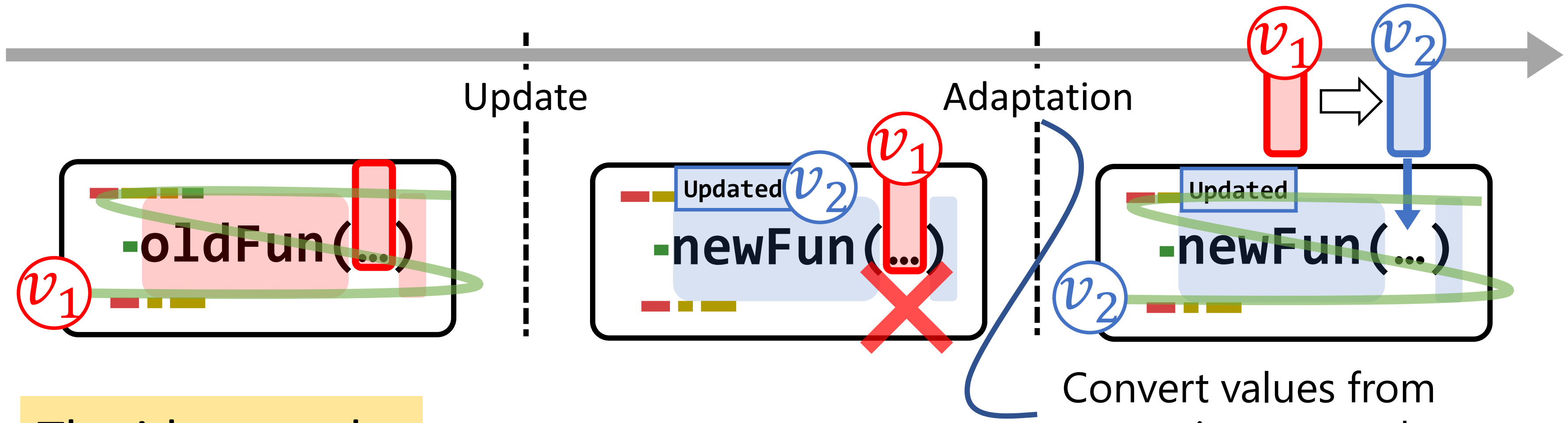
$\{l_1 : A, l_2 : B\}$

Allow different types across versions unlike λ_{VL}

Future Work

Automatic Adaptations

Type system can detect where adaptation is needed



The idea needs:

- **Code repository**, a *persistent definition/package store*
  - Working environment is regarded as ***view*** into the code repo.

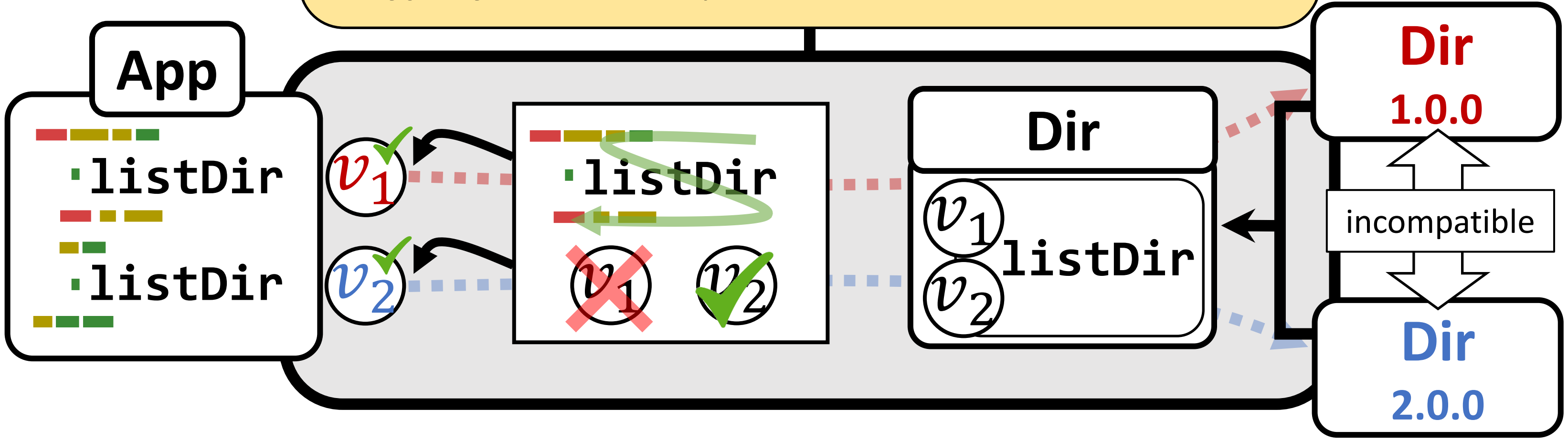
Nix [Dolstra'04]: Hashed packages + Nix manager
Unison: Hashed definitions + Unison codebase

+ ***Consistency checking within expressions***

Summary

Goal:
- *Using multiple versions* in one client
- *Detecting* inconsistent *version* within a program

Proposal: Core calculus $\lambda_{\mathbf{VL}}$
[COP'18, ‹Programming›'22]
- ***Versioned values*** to hold multiple versions of definitions
- ***Type system*** to analyze available versions



WIP
- Implementation
[COP'22, Doctoral Thesis]

Future work
- Integrating record calculus
- Automatic adaptation

Q&A

Typing Rules

$$\frac{}{\emptyset \vdash n : \mathrm{Int}}\ \text{int} \qquad \frac{}{x : A \vdash x : A}\ \text{var}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ \text{abs}$$

$$\frac{\Gamma_1 \vdash t_1 : A \to B \qquad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1\ t_2 : B}\ \text{app} \qquad \frac{\Gamma_1 \vdash t_1 : \Box_r A \quad \Gamma_2, x : [A]_r \vdash t_2 : B}{\Gamma_1 + \Gamma_2 \vdash \mathbf{let}\ [x] = t_1\ \mathbf{in}\ t_2 : B}\ \text{let}$$

$$\frac{\Gamma \vdash t : A}{\Gamma, [\Delta]_0 \vdash t : A}\ \text{weak} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A]_1 \vdash t : B}\ \text{der} \qquad \frac{[\Gamma] \vdash t : A}{r * [\Gamma] \vdash [t] : \Box_r A}\ \text{pr}$$

$$\frac{\Gamma, x : [A]_r, \Gamma' \vdash t : B \qquad r \sqsubseteq s}{\Gamma, x : [A]_s, \Gamma' \vdash t : B}\ \text{sub} \qquad \frac{\Gamma \vdash t : \Box_r A \quad l \in r}{\Gamma \vdash t.l : \Box_r A}\ \text{extr}$$

$$\frac{[\Gamma_i] \vdash t_i : A}{\bigcup(\{l_i\} * [\Gamma_i]) \vdash \langle \overline{l = t} \mid l_i \rangle : A}\ \text{veri} \qquad \frac{[\Gamma_i] \vdash t_i : A}{\bigcup(\{l_i\} * [\Gamma_i]) \vdash \{\overline{l = t} \mid l_i\} : \Box_{\{\bar{l}\}} A}\ \text{ver}$$

Q&A

Versioning Principle in VL

*Package developer put **different version***
if developer changes anything in the module

Supported incompatibilities

- Add/delete definitions
- Implementation change (with no type changes)
- Add/Delete imports

Maintain identity among different versions

Unsupported incompatibilities

- Type changes
- Module name changes

lose identity among different versions

(Unsupported features)

- Data type changes
- Type class changes
- License changes
- Publicity changes

Q&A

Intuition to 0 and 1 in Semiring

Both 0 and 1 indicate ***unavailable resources***.

Treated differently only in multiplication $\otimes$.

$$r_1 \otimes r_2 = \begin{cases} \bot & (r_1 = \bot \lor r_2 = \bot) \\ r_1 \cup r_2 & (otherwise) \end{cases}$$

$$\frac{\Gamma \vdash t : A}{\Gamma, [\Delta]_{\underset{=\bot}{0}} \vdash t : A}\text{weak} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma, x : [A]_{\underset{=\emptyset}{1}} \vdash t : B}\text{der}$$

$$\frac{\Gamma, x : [A]_r, \Gamma' \vdash t : B \quad r \sqsubseteq s}{\Gamma, x : [A]_s, \Gamma' \vdash t : B}\text{sub} \qquad \bot \sqsubseteq \emptyset \sqsubseteq \{l_i\} \sqsubseteq \cdots$$

In other coeffect calculi, the semantic difference between 0 and 1 may be meaningful.
i.e.) Exact usage $(\mathbb{N}, +, 0, \cdot, 1, \equiv)$[Patriceik'14,Orchard'19]