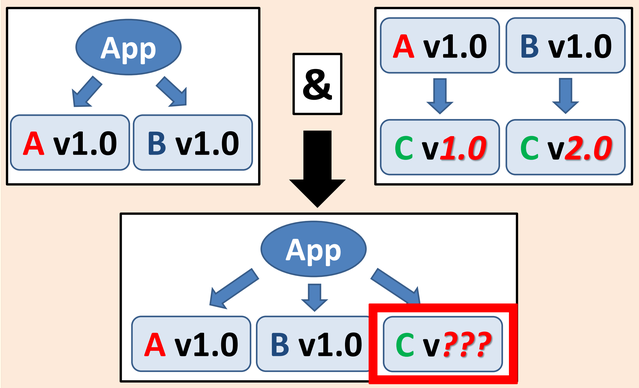
Graded Modality を用いた型ベースの互換性解析

田辺 裕大[1]　Luthfan Anshar Lubis[1]　青谷 知幸[2]　増原 英彦[1]（[1]東京工業大学　[2]豆蔵）

Problem

依存性地獄

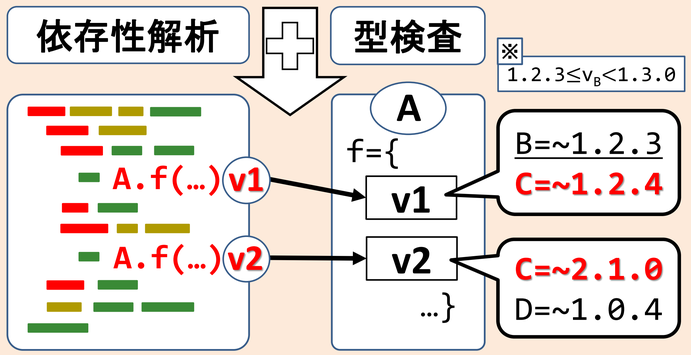
一般にライブラリの追加やアップデートは困難
（失敗の原因特定も困難）

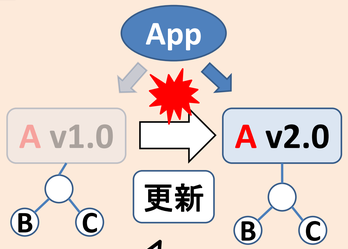


Proposal

型検査による互換性検査

① **複数Ver.の実装の共存**を許す言語設計
② 型検査で**式単位で依存するバージョンを解析**
③ 型検査で得た**制約に応じた実装を使用**





[Warning]
B.f は非推奨
C.g は動作変更

Idea

Versions as Computational Resources

*Coeffect*計算：計算的資源の使用状況を
追跡可能な型システム

Resources	Modalities (Coeffects)	Examples
Linearity[Brunel 14]	値の使用回数	0, 1
Security[Orchard 19]	Accessibility	Private, Public
Version[Tanabe 18]	**使用可能なVer.の集合**	{V1}, {V1,V2},..

データ型昇格[Yorgey 12]

Ver.違いは
GADT+型引数で表現

```
Ver = V1 | V2
data Vec (v:Ver) (t:Type) where
Nil  :: Vec V1 a;
Cons :: a → Vec V1 a → Vec V1 a
Arr  :: int -> (int->a) -> Vec V2 a
```

```
length:∀{v:Ver, t:Type}.
    {v=V1||v=V2} => Vec v t -> Int
length Nil = 0;
length Cons x xs = 1 + (length xs)
length Arr n init = n
```

Granuleへ
コンパイル

型制約にエンコード

Code Examples

```
vec1 : Vec@V1 Int
vec2 : Vec@V2 Int
vec3 : Vec@V3 Int
```

```
➢ length vec1/vec2
 - well-typed!
➢ length vec3
 - ill-typed!
➢ append vec1 vec2
 - ill-typed!
```

返り値もバージョン制限有り
vを介してバージョンの制約を伝播

```
append : Vec@v t → Vec@v t → Vec@v t
  context v of -- バージョンのパターンマッチ
    (V1) ->
      append Nil ys = ys
      append (Cons x xs) ys = Cons x …
    (V2) ->
      append (Arr m f) (Arr n g) = …
```

V1
```
data Vec (t:Type) where
-- Listで実装
```

V2
```
data Vec (t:Type) where
-- Arrayで実装
```

```
length : Vec@v t → Int
  context v of -- バージョンのパターンマッチ
    (V1) ->
      length Nil = 0
      length Cons x xs = 1 + (length xs)
    (V2) ->
      length Arr n init = n
```

Granule

*Coeffect*計算をコアに持つ言語[Orchard 18]

【特徴1】Indexed Type（軽量依存型）

```
append:∀{t:Type, n,m:Nat}.
  Vec n t → Vec m t → Vec (n+m) t
append Nil ys = ys
append (Cons x xs) ys = Cons x (append xs ys)
```

```
data N (n:Nat) where
  Z : N 0;
  S : N n → N (n+1)
```

【特徴2】*Coeffect*を用いた型制約
（パラメータ多相＋双方向型検査[Dunfield 13]＋SMTソルバ）

```
sub:∀m n.{m > n} ⇒ N m → N n → N (m-n)
sub m Z = m
sub (S m) (S n) = sub m n
sub Z (S n) = sub Z n -- {m > n}に違反
```

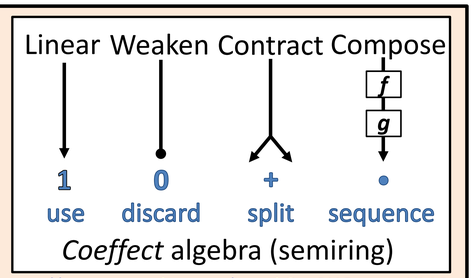
rejected

Type System

半環 $(R, +, 0, \times, 1, \leq)$
＋
一般化有界線形論理

【1】型を$r \in R$で修飾

$$\Gamma, x{:}\,[T]_r \vdash t : T$$



図：https://www.youtube.com/watch?v=2HOtpcrmXMQ

【2】通常の型付けと共に資源管理を行う型規則
（プログラムの使われ方次第でrが決定）

$$\frac{\Gamma, x{:}\,A \vdash t : B}{\Gamma, x{:}\,[A]_1 \vdash t : B}\ \text{Der} \qquad \frac{\Gamma_1 \vdash t_1 : A \to B \quad \Gamma_2 \vdash t_2 : A}{\Gamma_1 + \Gamma_2 \vdash t_1\ t_2 : B}\ \text{App}$$

Future Work

- Semantic Versioning[Preston-Werner 2013]制約の型表現
  - Ver.のパターンマッチをSemVer制約式にしたい
  - ~1.2.3, ^1.2.3, 1.2.* ― 順序が定義できれば十分？
- 軽量依存型システムを持つ言語へのコンパイル方法
  - GADT + Granule + データ型昇格？
  - （普通の）依存型を持つ他言語？

References

- [Yorgey 12] Giving Haskell a Promotion. (TLDI '12)
- [Brunel 14] A Core Quantitative Coeffect Calculus. (ESOP '14)
- [Orchard 19] Quantitative Program Reasoning with Graded Modal Types. (ICFP '19)
- [Tanabe 18] A Context-Oriented Programming Approach to Dependency Hell. (COP '18)
- [Dunfield 13] Complete and easy bidirectional typechecking for higher-rank polymorphism. (ICFP '13)