スマートコントラクトのガス消費量のResource Aware MLを用いた静的解析

2021/2/10

五十嵐・末永研究室
小野　雄登

研究背景：スマートコントラクト

- **ブロックチェーン上で動作するプログラム**
  - **仮想通貨の取引における計算などを自動化**

- **ブロックチェーンごとにスマートコントラクトの記述言語がある**

  例）Ethereum : Solidity

  Tezos : Michelson, SCaml

研究背景：ガス

- スマートコントラクトが利用する計算資源にかかる手数料

- ガス消費量を静的に解析する手法が求められている
  - ガス欠→コントラクトの実行中止 & 無駄な手数料の発生

本研究：Tezosにおけるガス消費量の静的解析

- **入力：Michelsonで記述されたスマートコントラクト**
- **出力：ガス消費量の見積もり**

方針

1. **MichelsonプログラムをResource Aware ML（RAML）[Jan Hoffmann, et al. CAV 2012] でエンコード**
2. **エンコードしたプログラムを解析することでガス消費量の見積もりを計算**

Resource Aware ML (RAML)

- OCaml文法を備えた関数型プログラミング言語
  - プログラムのリソース消費量解析の機能を備える
  - 様々なメトリックに基づくリソース消費量の解析が可能

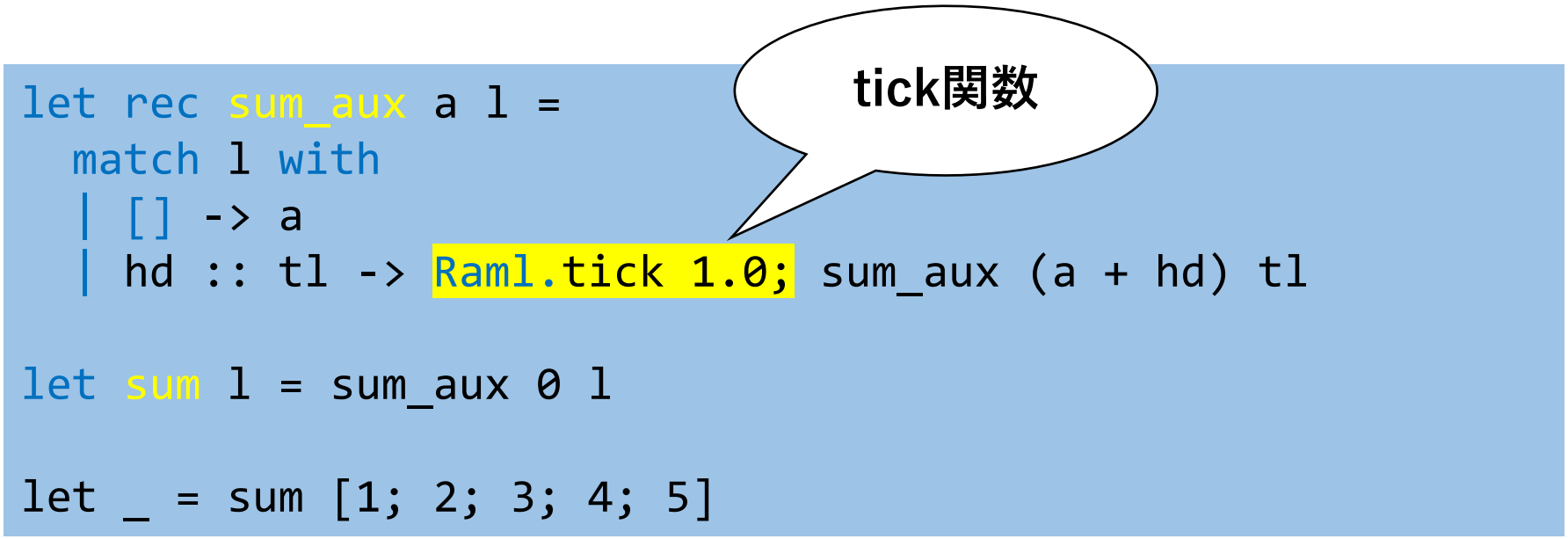
    例) 計算ステップ数，メモリ消費量，...

- 本研究で用いるメトリック：tick
  - ユーザーが定義したリソース消費量を解析する
  - プログラム中でtick関数を呼び出すことで，関数のリソース消費を表現

Resource Aware ML (RAML)

```
let rec sum_aux a l =
  match l with
  | [] -> a
  | hd :: tl -> Raml.tick 1.0; sum_aux (a + hd) tl

let sum l = sum_aux 0 l

let _ = sum [1; 2; 3; 4; 5]
```

Resource Aware ML (RAML)



Resource Aware ML (RAML)

```
let rec sum_aux a l =
  match l with
  | [] -> a
  | hd :: tl -> Raml.tick 1.0; sum_aux (a + hd) tl

let sum l = sum_aux 0 l

let _ = sum [1; 2; 3; 4; 5]
```

Resource Aware ML (RAML)

```
$ ./main analyze ticks 1 4 -print level 1 sum.raml

  Trying degree: 1

  Function types:

== sum :

  int list -> int

  Simplified bound:
     1*M
   where
     M is the number of ::-nodes of the argument

====

  Derived upper bound: 5.00
```

Resource Aware ML (RAML)

```
$ ./main analyze ticks 1 4 -print level 1 sum.raml

  Tryi
  Funct

== sum :

  int list -> int

  Simplified bound:
     1*M
   where
     M is the number of ::-nodes of the argument

====

  Derived upper bound: 5.00
```

メトリック

次数

Resource Aware ML (RAML)

```
$ ./main analyze ticks 1 4 -print level 1 sum.raml

  Trying degree: 1

  Function types:

== sum :

  int list

  Simplified bound:
     1*M
   where
     M is the number of ::-nodes of the argument

====

  Derived upper bound: 5.00
```

関数のリソース多項式

Resource Aware ML (RAML)

```
$ ./main analyze ticks 1 4 -print level 1 sum.raml

  Trying degree: 1

  Function types:

== sum :

  int list -> int

  Simplified bound:
     1*M
   where
     M is the number of ::-n

====

  Derived upper bound: 5.00
```

リソース消費量の上界

Michelsonプログラムの例

Michelson : スタックベースのプログラミング言語

```
parameter int ;
storage int ;
code { CAR ;
       DUP ;
       ADD ;
       NIL operation ;
       PAIR ; }
```

Michelsonプログラムの例

Michelson : スタックベースのプログラミング言語

```
parameter int ;
storage int ;
code { CAR ;
       DUP ;
       ADD ;
       NIL operation ;
       PAIR ; }
```

型宣言

命令列

各命令はスタックを書き換える

Michelsonプログラムの例

```
parameter int ;
storage int ;
code { CAR ;
       DUP ;
       ADD ;
       NIL operation ;
       PAIR ; }
```

int 3
int 3

Michelsonプログラムの例

```
parameter int ;
storage int ;
code { CAR ;
       DUP ;
       ADD ;
       NIL operation ;
       PAIR ; }
```

int 6

Michelson命令のエンコード

- スタックの要素：ヴァリアント型t

```
type t = Int of int | Bool of bool | Pair of t * t | …
```

- スタック：型tのリスト
- 命令：(t list → t list)型の関数

```
let add s =
  match s with
  | Int x :: Int y :: xs -> Raml.tick 4.0; Int (x + y) :: xs
  | Int x :: Nat y :: xs -> …
    ⋮
  | _ -> raise Invalid argument
```

Michelsonプログラムのエンコード

- 設計したライブラリを用いて，Michelsonプログラムを**RAML**プログラムにエンコードできる

```
parameter int ;
storage int ;
code { CAR ;
       DUP ;
       ADD ;
       NIL operation ;
       PAIR ; }
```

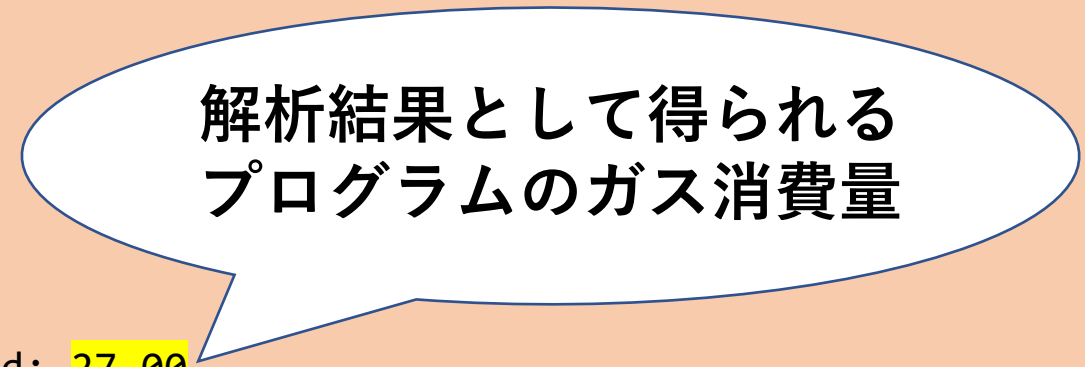
Michelsonプログラム

```
let _ =
 (pair
 (nil
 (add
 (dup
 (car
 (Pair (Int 3, Int 0) :: [])))))))
```

RAMLプログラム

エンコードしたプログラムの解析

```
$ ./main analyze ticks 1 4 -print level 1 example.raml

  Trying degree: 1

  Function types:

== pair :

  t list -> t list

  Simplified bound:
     8
====

  Derived upper bound: 27.00
```



解析例

プログラム	実際のコスト	RAMLの見積もり
example1	43	37
example2	282	225
example3	28	28
example4	80	failed
example5	12079	12079

- example1 : 整数値演算
- example2 : 整数値演算，ループ
- example3 : 条件分岐
- example4 : 整数値演算，リストに対する再帰
- example5 : コントラクトの生成，通貨の送金

解析例

プログラム		り
example1		
example2	282	225
example3	28	28
example4	80	failed
example5	12079	12079

ガス消費量を正しく
見積もることができた

- example1 : 整数値演算
- example2 : 整数値演算，ループ
- example3 : 条件分岐
- example4 : 整数値演算，リストに対する再帰
- example5 : コントラクトの生成，通貨の送金

解析例

プログラム	実際のコスト	RAMLの見積もり
example1	43	37
example2	282	225
example3	28	28
example4		
example5		

整数値演算を行う命令を含むコントラクトでは，見積もりに多少の誤差が生じた

- example1 : 整数値演算
- example2 : 整数値演算，ループ
- example3 : 条件分岐
- example4 : 整数値演算，リストに対する再帰
- example5 : コントラクトの生成，通貨の送金

解析例

プログラム	実[illegible]	見積もり
example1	[illegible]	[illegible]
example2	[illegible]	[illegible]
example3	28	28
example4	80	failed
example5	12079	12079

リストの再帰を行う命令を含むプログラムは，解析が行えなかった

- example1 : 整数値演算
- example2 : 整数値演算，ループ
- example3 : 条件分岐
- example4 : 整数値演算，リストに対する再帰
- example5 : コントラクトの生成，通貨の送金

まとめ

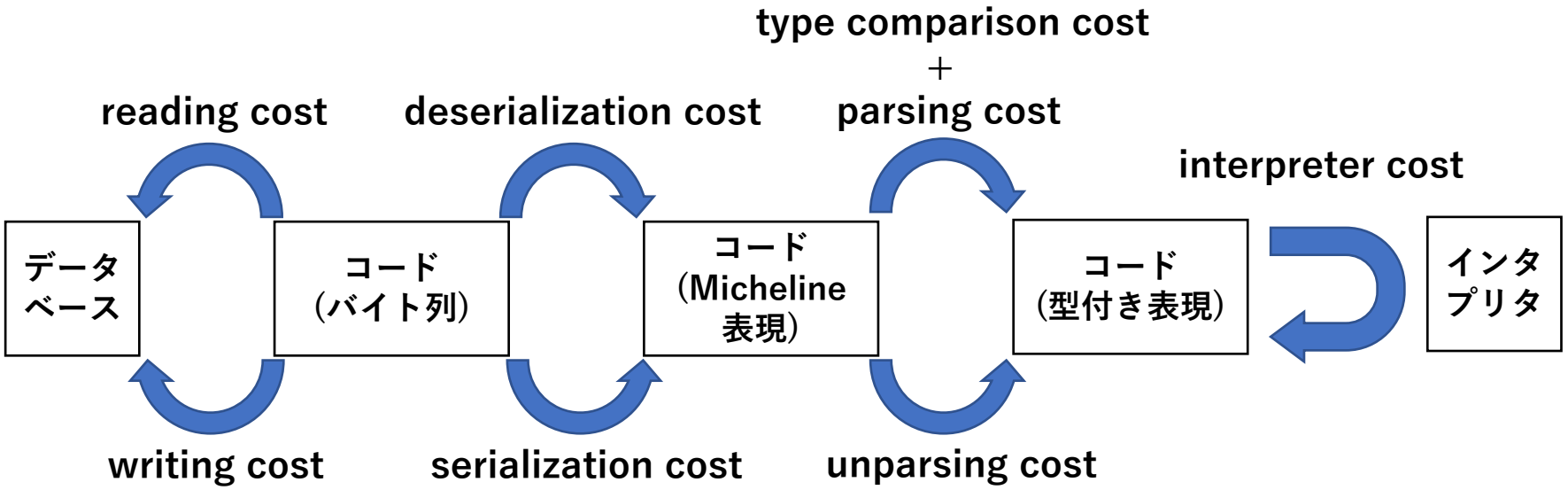
- Michelsonの各命令を模倣するライブラリをRAMLで作成し，ライブラリを用いてMichelsonプログラムをRAMLでエンコードした.

- エンコードしたRAMLプログラムをtickメトリックを用いて解析することで，コントラクトのガス消費量を見積もった.

- ライブラリの拡張，より正確なガス消費量の見積もりなどが今後の課題として挙げられる.

付録：Tezosにおけるガス消費の仕組み

- ガス消費：コントラクトの実行段階によって8つのコストに分けられる



- 本研究では特にinterpreter costの見積もりを目的とした