確率的プログラミング言語の形式基盤を構文で拡張する試み

斉藤 歩夢 [1]　Affeldt Reynald [2]

[1]東京工業大学 情報理工学院 数理・計算科学系　[2]産業技術総合研究所 (AIST)

形式化中：確率的プログラミング言語

- **確率的プログラミング言語の例（擬似コード）**
  平日は10台/h、週末は3台/hバスが通る地点で1時間に4台バスを観測したとき、その日が週末である確率を表すモデル

```
normalize (
  let x = sample (bernoulli (2 / 7)) in
  let r = if x then 3 else 10 in
  let _ = score (r ^ 4 / 4! * e ^ (- r)) in
  return x)
```

  1. 事前確率からサンプリング(sample)
  2. その値に応じてパラメータを設定
  3. 与えられたパラメータの尤度を評価(score)
  4. 尤度を正規化し、事後分布を返す(normalize)
- **意味論**
  (ポアソン分布$poisson(\lambda = 4)$においてr=3となる尤度) $= \frac{3^4}{4!}e^{-3}$

$$\frac{1}{N}\left(\frac{2}{7}\times\frac{3^4}{4!}e^{-3}\delta_T + \frac{5}{7}\times\frac{10^4}{4!}e^{-10}\delta_F\right) \approx 0.78\delta_T + 0.22\delta_F$$

$(0.78\delta_T + 0.22\delta_F)$：確率測度

  - 各命令は主にs-有限核
  - 核 $X \rightsquigarrow Y$ は関数$k : X \to \underbrace{\Sigma_Y \to [0,\infty]}_{\text{測度}}$

    ただし、$\forall U \in \Sigma_Y,\ x \mapsto k\,x\,U$ は 可測関数
- Coq→MathComp→MathComp-Analysis[1] → prob_lang.v + lang_syntax.v[5]
  MathComp-Analysisライブラリの測度論やルベーグ積分の形式化を活用し拡張

意味論におけるs-有限核の合成

- s-有限核：有限核の可算無限和として表される核
  - 有限核： すべてのxについて$k\ x\ U$が有限値をとるような核k
- s-有限核の合成は安定的である[6]

$$l : X \rightsquigarrow Y \qquad k : X \times Y \rightsquigarrow Z$$

$$l \fatsemi k : X \rightsquigarrow Z$$

$$\stackrel{def}{=} \lambda x\,U.\ \int_y k\,(x,y)\,U(\mathbf{d}\,l\,x)$$

$[\![$let x := t in u$]\!]$はs-有限核の合成として与えられる

$$[\![\mathtt{t}]\!] : [\![\Gamma]\!] \xrightarrow{\text{s-fin}} [\![\mathbf{A}]\!] \qquad [\![\mathtt{u}]\!] : [\![\Gamma; x : \mathbf{A}]\!] \xrightarrow{\text{s-fin}} [\![\mathbf{B}]\!]$$

$$[\![\mathtt{let}\ x{:=}\mathtt{t}\ \mathtt{in}\ \mathtt{u}]\!] : [\![\Gamma]\!] \xrightarrow{\text{s-fin}} [\![\mathbf{B}]\!]$$

$$\stackrel{def}{=} [\![\mathtt{t}]\!] \fatsemi [\![\mathtt{u}]\!]$$

- Coqでの意味論の上での形式化 [2]

```
Definition kstaton_bus : R.-sfker T ~> bool :=
  letin (sample (bernoulli p27))
    (letin (ite var2of2 (ret k3) (ret k10))
      (letin (score (measurable_fun_comp mh var3of3))
  (ret var2of4))).
Definition staton_bus := normalize kstaton_bus.
```

  var2of2, var3of3などは環境にアクセスするための関数
- **問題点：**
  1. 構文的性質を形式化するのが困難
  2. コード生成が困難

一階確率的プログラミング言語の構文

- **型**

```
Inductive stype :=
| sunit | sbool | sreal
| spair : stype -> stype -> stype
| sprob : stype -> stype (* 確率分布の型を表す（例えば、sprob sreal は実数上の確率分布） *)
| sproduct : list stype -> stype.
```

- **環境**
  $\Gamma; x : \mathbf{A}$は直積型として定義
- **項：決定項(expD) + 確率項(expP)**
  - Intrinsically-typed syntaxとして定義[3, 4]

```
Inductive expD : context -> stype -> Type :=
| exp_unit l : expD l sunit
| ...
| exp_pair l t1 t2 : expD l t1 -> expD l t2 -> expD l (spair t1 t2)
| exp_var l x t : t = nth sunit (map snd l) (seq.index x (map fst l)) ->
    expD l t
| exp_norm l t : expP l t -> expD l (sprob t)
...( 省略 )...
with expP : context -> stype -> Type :=
| exp_if l t : expD l sbool -> expP l t -> expP l t -> expP l t
| exp_return l t : expD l t -> expP l t
| exp_letin l l' t1 t2 (x : string) : l' = (x, t1) :: l ->
    expP l t1 -> expP l' t2 -> expP l t2
...( 省略 )... .
```

評価関係eval

- **構文と意味論の対応**

構文		意味論	
決定項	$\Gamma \vdash_d t : \mathbf{A}$	可測関数	$[\![\Gamma]\!] \longrightarrow [\![\mathbf{A}]\!]$
閉じた決定項	$\vdash_d t : \mathbf{A}$	$[\![\mathbf{A}]\!]$上の測度	
確率項	$\Gamma \vdash_p t : \mathbf{A}$	s-有限核	$[\![\Gamma]\!] \xrightarrow{\text{s-fin}} [\![\mathbf{A}]\!]$

- **構文と意味論の対応関係をInductiveで定義**

```
Inductive evalD : forall (l : context) (T : stype) (e : @expD R l T)
  (f : projT2 (typei (sprod (map snd l))) -> projT2 (typei T)),
  measurable_fun setT f -> Prop :=
| E_real : l |- exp_real r -D-> cst r # kr r (* kr r は (cst r)が可測関数であることの証明 *)
...( 省略 )...
with evalP : forall (l : context) (T : stype),
  expP l T ->
  R.-sfker (projT2 (typei (sprod (map snd l)))) ~> projT2 (typei T) -> Prop := ...( 省略 )... .
```

  - typei : stypeを受け取り可測空間を返す関数
  - 決定項 t : $\Gamma \vdash t \to_D f \,\#\, mf$ (mfはfが可測関数であることの証明)
  - 確率項 t : $\Gamma \vdash t \to_P k$
- **関係evalの関数的性質の証明**
  - **一意性**

```
Lemma evalD_uniq l t f g mf mg:
  l |- t -D-> f # mf ->
  l |- t -D-> g # mg ->
  f = g.
```

  - **全域性**

```
Lemma evalD_full l :
  forall t, {subset (FV t) <= map fst l} ->
  exists f mf, l |- t -D-> f # mf.
```

  確率項に関する関係についても同様の補題を示した

実験： Let-in命令の交換法則の証明

- **目標の補題** letinC :

$$x \notin \mathrm{FV(u)}, y \notin \mathrm{FV(t)} \Rightarrow \left[\!\!\left[\begin{array}{l}\mathtt{Let\ x\ <-\ t\ in}\\ \mathtt{Let\ y\ <-\ u\ in}\\ \mathtt{Ret\ (x,\ y)}\end{array}\right]\!\!\right] = \left[\!\!\left[\begin{array}{l}\mathtt{Let\ y\ <-\ u\ in}\\ \mathtt{Let\ x\ <-\ t\ in}\\ \mathtt{Ret\ (x,\ y)}\end{array}\right]\!\!\right]$$

  (FV(t)は項tに含まれる自由変数)
- **補題** letinC **において**t = Ret 1, u = Ret 2 **とした補題** letinC12:

$$\left[\!\!\left[\begin{array}{l}\mathtt{Let\ x\ <-\ Ret\ 1\ in}\\ \mathtt{Let\ y\ <-\ Ret\ 2\ in}\\ \mathtt{Ret\ (x,\ y)}\end{array}\right]\!\!\right] = \left[\!\!\left[\begin{array}{l}\mathtt{Let\ y\ <-\ Ret\ 2\ in}\\ \mathtt{Let\ x\ <-\ Ret\ 1\ in}\\ \mathtt{Ret\ (x,\ y)}\end{array}\right]\!\!\right]$$ の証明が完了した
  - 評価関係evalとその関数的性質を使用
    → s-有限測度に関するフビニの定理を適用できる形にして証明

```
Variable (m1 : {sfinite_measure set X -> \bar R}).
Variable (m2 : {sfinite_measure set Y -> \bar R}).
Lemma sfinite_fubini f :
\int[m1]_x \int[m2]_y f (x, y) = \int[m2]_y \int[m1]_x f (x, y).
```

- **次の課題：補題**letinC, **補題**letinA**（let-inの結合法則）**

  letinA :

$$[\![\mathtt{Let\ x\ <-\ t\ in\ Let\ y\ <-\ u\ in\ v}]\!] = [\![\mathtt{Let\ y\ <-\ (Let\ x\ <-\ t\ in\ u)\ in\ v}]\!]$$

関連研究

[1] Reynald Affeldt, Yves Bertot, Cyril Cohen Marie Kerjean, Assia Mahboubi, Damien Rouhling, Pierre Roux, Kazuhiko Sakaguchi, Zachary Stone, Pierre-Yves Strub, and Laurent Théry.
MathComp-Analysis: Mathematical components compliant analysis library.
https://github.com/math-comp/analysis, 2022.
Since 2017. Version 0.5.4.

[2] Reynald Affeldt, Cyril Cohen, and Ayumu Saito.
Semantics of probabilistic programs using s-finite kernels in coq.
In *Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2023, page 3–16, New York, NY, USA, 2023. Association for Computing Machinery.

[3] Reynald Affeldt and Kazuhiko Sakaguchi.
An intrinsic encoding of a subset of c and its application to tls network packet processing.
Journal of Formalized Reasoning, 7(1):63–104, 2014.

[4] Casper Bach Poulsen, Arjen Rouvoet, Andrew Tolmach, Robbert Krebbers, and Eelco Visser.
Intrinsically-typed definitional interpreters for imperative languages.
Proc. ACM Program. Lang., 2(POPL), dec 2017.

[5] Ayumu Saito.
MathComp-Analysis at kernel-syntax, 2023.
https://github.com/AyumuSaito/analysis/tree/kernel-syntax/theories.

[6] Sam Staton.
Commutative semantics for probabilistic programming.
In *26th European Symposium on Programming (ESOP 2017), Uppsala, Sweden, April 22–29, 2017*, volume 10201 of *Lecture Notes in Computer Science*, pages 855–879. Springer, 2017.

Future work

- 環境を表すリストを完全に省略可能な引数とする
- Custom entryを用いることでより簡潔に記述することができないか試みる
- 実際の確率的プログラミング言語でコンパイル可能なコードを生成する

謝辞

本研究はJSPS科研費 JP22H00520の助成を受けたものです。