AIPL への型推論の導入と型健全性の証明

AIPL (Array-Inspired Pipeline Language) ランク階層フラグメント

2026 年 6 月 3 日

概要

配列指向パイプライン言語 AIPL に対し，**型推論**を導入する．具体的には，型健全性が成り立つ中核フラグメント（ランクで層別された入れ子配列と整数ランクの演算子からなる部分言語）の構文を定義し，型（ = ランク）システムを与え，**進行・保存・型安全性・決定性**を証明する．すべての定理は The Rocq Prover (Coq) 9.1.0 によって機械検証済みである（`AIPL.v`）．

目次

1 **前提と但し書き** 1

2 **AIPL の意味論（公開モデルの要約）** 1

3 **健全性が成り立つ部分言語（フラグメント）の構文** 2

4 **型システム（型 = ランク）** 2

5 **操作的意味論（スモールステップ）** 3

6 **型健全性の証明** 3

7 **Coq による機械化（自動証明）** 4

8 **まとめ** 6

1 前提と但し書き

ご指定の論文は Gmail の添付ファイル URL で共有されていたが，この URL は Google ログインを要求するため内容を取得できなかった（認証が必要なページはフェッチできない）．そこでご了承いただいた方針に従い，**AIPL の公開実行モデル**（GitHub: `saulpw/aipl` のドキュメント）を意味論の出発点として形式化した．

注意．したがって本書の健全性は「公開モデルから再構成した操作的意味論」に対するものである．論文中の意味論が細部で異なる場合は定義を論文に合わせて差し替える必要があるが，その際も証明構造（補題 1 + 進行 + 保存）はほぼそのまま再利用できる．論文本文を共有いただければ，その意味論に厳密に合わせて再証明する．

2　AIPL の意味論（公開モデルの要約）

AIPL は「配列指向のパイプライン言語」であり，次の特徴を持つ．

- **データモデル:** 中心構造は **Table**（行 ＝ ハッシュマップの配列，列 ＝ 名前付きキー）．各テーブルの**一番右の列が主値**を表す．
- **ランク (Rank):** 主値列の入れ子の深さがランクを決める．ランク 0 はスカラのベクトル，ランク $n+1$ は最右列がさらにテーブルになっているもの（入れ子が 1 段増えるとランクが 1 増える）．
- **演算子 (operator):** 各演算子は入力ランク `rankin` と出力ランク `rankout` を持つ．
- **暗黙のループ (tacit dataflow):** 演算子は「最も深く入れ子になったテーブル」に自動的に適用され，結果が次の演算子の入力になる．明示的ループなしに全要素へ反復適用される．

このランク多相は，APL 系言語の**セル (cell) とフレーム (frame)** の考え方そのものである．ランク r の値に `rankin` の演算子を適用するとき，

$$\text{セルランク} = \text{rankin}, \qquad \text{フレームランク} = r - \text{rankin}$$

であり，結果のランクは $(r - \text{rankin}) + \text{rankout}$ になる．

■つまずき（stuck）はどこで起きるか．「値のランク r が演算子の要求する rankin より小さい」とき，演算子は処理すべきセルを見つけられず**未定義（stuck）**になる．これが型システムで防ぐべき唯一の動的エラーである．

3　健全性が成り立つ部分言語（フラグメント）の構文

完全な AIPL には健全性を素朴には保証できない機能（動的な列名解決，半端ランク 0.5/1.5，副作用 `!llm` 等）が含まれる．そこで**型健全性が成り立つ核**だけを切り出す．

除外する機能	除外理由
名前付き列・異種行・親行での名前解決	動的な列名参照は「列が無い」で stuck しうる．静的健全性にはレコード型が別途必要．本フラグメントは主値スパインのみ保持
半端ランク 0.5（行），1.5（テーブル）	行・テーブル形状の型付けが必要．整数ランク（セル/フレーム）に限定
副作用（`!llm`, fetch 等）	ランク健全性とは独立．純粋な値変換 denote として抽象化

表 1　フラグメントから除外する機能と理由．この絞り込みが「健全性が成り立つ部分の構文を定義する」ことの中身である．

■構文の定義. **値（*value*）**はスカラまたは値のベクトル:

$$v ::= \mathsf{Scalar}\ b \mid \mathsf{Vec}\ [v_1, \ldots, v_m] \qquad (b \in \mathsf{Base})$$

演算子（*operator*） o は，入力ランク $\mathrm{rin}(o) \in \mathbb{N}$，出力ランク $\mathrm{rout}(o) \in \mathbb{N}$，および 1 セルに対する原始的作用 $\mathrm{denote}(o)$ を持つ．**パイプライン（*pipeline*）**は演算子の列:

$$p ::= [\,] \mid o :: p$$

4　型システム（型 ＝ ランク）

本フラグメントでは**型はランク（自然数）**そのものである．型推論は「入力ランクから各段の出力ランクを伝播させる」だけで完結する．

■値のランク付け $\mathrm{has\_rank}\ n\ v$.

$$\frac{}{\mathrm{has\_rank}\ 0\ (\mathsf{Scalar}\ b)}\ (\text{HR-Scalar}) \qquad \frac{\mathrm{Forall}\,(\mathrm{has\_rank}\ n)\ vs}{\mathrm{has\_rank}\ (n{+}1)\ (\mathsf{Vec}\ vs)}\ (\text{HR-Vec})$$

スカラはランク 0．ベクトルは全要素が同一ランク n のときランク $n+1$.

■パイプラインの型付け $\mathrm{ptype}\ p\ r\ r'$. 「パイプライン p は入力ランク r の値を出力ランク r' の値に変換する」という判断:

$$\frac{}{\mathrm{ptype}\ [\,]\ r\ r}\ (\text{PT-Nil})$$

$$\frac{\mathrm{rin}(o) \le r \qquad \mathrm{ptype}\ p\ \big((r - \mathrm{rin}(o)) + \mathrm{rout}(o)\big)\ r'}{\mathrm{ptype}\ (o :: p)\ r\ r'}\ (\text{PT-Cons})$$

前提 $\mathrm{rin}(o) \le r$ が**静的な健全性条件**である．これが満たされていれば演算子はフレームランク $r - \mathrm{rin}(o)$ の暗黙ループの底でちょうどセルを見つけられ，stuck しない．次段へ渡るランクは $(r - \mathrm{rin}(o)) + \mathrm{rout}(o)$.

注意．この規則は**そのまま型推論アルゴリズム**である．入力ランク r を与えると，各段で $r \leftarrow (r - \mathrm{rin}(o)) + \mathrm{rout}(o)$ と更新しつつ $\mathrm{rin}(o) \le r$ を検査するだけ（線形時間・決定的）.

5　操作的意味論（スモールステップ）

実行時状態（configuration）を「現在のランク r，値 v，残りパイプライン p」の三つ組とする．

■演算子適用の本体. フレーム深さ $k = r - \mathrm{rin}(o)$ だけ潜って各セルに $\mathrm{denote}(o)$ を適用する:

$$\mathrm{applyOp}\ o\ 0\ v = \mathrm{denote}(o)\ v, \qquad \mathrm{applyOp}\ o\ (k{+}1)\ (\mathsf{Vec}\ vs) = \mathsf{Vec}\,(\mathrm{map}\,(\mathrm{applyOp}\ o\ k)\ vs)$$

■ステップ規則.

$$\frac{\mathrm{rin}(o) \le r}{\langle r,\ v,\ o :: p\rangle \ \longrightarrow\ \big\langle (r - \mathrm{rin}(o)) + \mathrm{rout}(o),\ \mathrm{applyOp}\ o\ (r - \mathrm{rin}(o))\ v,\ p\big\rangle}\ (\text{St-Op})$$

終了状態: $\langle r, v, [\,]\rangle$（残りパイプラインが空 ＝ 値が得られた）． **stuck:** $\langle r, v, o :: p\rangle$ で $\mathrm{rin}(o) > r$ のとき.

■原始作用への唯一の仮定.

$$(\text{DENOTE\_RANK}) \quad \text{has\_rank}\ (\text{rin}(o))\ v \implies \text{has\_rank}\ (\text{rout}(o))\ (\text{denote}(o)\ v)$$

6 型健全性の証明

補題 1 (applyOp のランク保存)**.** has_rank $(k{+}\text{rin}(o))\ v \Rightarrow$ has_rank $(k{+}\text{rout}(o))$ (applyOp $o\ k\ v$).

Proof. k に関する帰納法. $k = 0$**:** 仮定は has_rank $(\text{rin}\, o)\ v$. DENOTE_RANK より has_rank $(\text{rout}\, o)$ $(\text{denote}\, o\, v)$. $k + 1$**:** 仮定 has_rank $(S(k + \text{rin}\, o))\ v$ を反転すると $v = \mathsf{Vec}\ vs$ かつ Forall $(\text{has\_rank}\, (k + \text{rin}\, o))\ vs$. 帰納法の仮定を各要素に適用すると Forall $(\text{has\_rank}\, (k + \text{rout}\, o))\, (\text{map}\, (\text{applyOp}\, o\, k)\, vs)$, HR-Vec より結論. □

定理 1 (進行 / Progress)**.** ptype $p\ r\ r'$ かつ has_rank $r\ v$ ならば, $p = [\,]$（終了）か, ある状態へ *1* ステップ進める. ***stuck* しない.**

Proof. ptype の導出で場合分け. PT-Nil なら $p = [\,]$. PT-Cons なら $\text{rin}(o) \le r$ が得られ St-Op が適用可能. □

定理 2 (保存 / Preservation)**.** ptype $(o :: p)\ r\ r'$, has_rank $r\ v$, かつ $\langle r, v, o :: p\rangle \to \langle r_2, v_2, p\rangle$ ならば, ptype $p\ r_2\ r'$ かつ has_rank $r_2\ v_2$.

Proof. PT-Cons の反転より $r_2 = (r - \text{rin}\, o) + \text{rout}\, o$ かつ ptype $p\ r_2\ r'$. 値については $\text{rin}(o) \le r$ より $(r - \text{rin}\, o) + \text{rin}\, o = r$ なので, 補題 1（$k = r - \text{rin}\, o$）から has_rank $r_2\ v_2$. □

定理 3 (型安全性 / Type Safety（主定理）)**.** ptype $p\ r\ r'$ かつ has_rank $r\ v$ ならば, ある値 v_f が存在して

$$\langle r, v, p\rangle \longrightarrow^* \langle r', v_f, [\,]\rangle \qquad \text{かつ} \qquad \text{has\_rank}\ r'\ v_f.$$

Proof. p の構造帰納法. 空なら PT-Nil より $r' = r$, $v_f = v$. $o :: p$ なら St-Op で 1 ステップ進め（進行）, 保存により不変条件が保たれ, 帰納法の仮定で残りを評価する. 各ステップでパイプライン長が 1 減るので有限ステップで必ず終了する（停止性も同時に従う）. □

注意. 各ステップで演算子を 1 個消費するため本フラグメントは**強正規化**する. well-typed なパイプラインは「決して stuck せず, 必ず予測どおりの出力ランクの値に評価される」.

定理 4 (決定性 / Determinism)**.** 任意の状態は高々 *1* つの後続状態へしか進まない（意味論は関数）.

7 Coq による機械化（自動証明）

上記すべてを `AIPL.v` に形式化し, **Rocq Prover (Coq) 9.1.0 でコンパイル成功（エラー・警告なし, 終了コード 0）**を確認済みである.

■自動化の方針.

紙の上の定義／定理	Coq の名称
値 v	`Inductive val` (Scalar/Vec)
ラ ン ク 付 け has_rank	`Inductive has_rank`
演 算 子 の 署 名 rin, rout, denote	`Variable rin / rout / denote`
原始作用のランク整合 DENOTE_RANK	`Hypothesis denote_rank`
applyOp	`Fixpoint applyOp`
補題 1	`Lemma applyOp_rank`
パイプライン型付け ptype	`Inductive ptype`
ステップ／多段ステップ	`Inductive step / multistep`
進行・保存・型安全性・決定性	`progress / preservation / type_safety / step_deterministic`

表 2　紙の上の定義・定理と Coq 形式化の対応.

- 演算子の集合・署名・原始作用は Section の `Variable` ／ `Hypothesis` として抽象化．全定理が「`denote_rank` を満たす任意の演算子系」に対して成立し，Section を閉じると自動で一般化される．
- 証明は `induction` ＋ `inversion` ＋ `constructor`/`eapply` に加え，算術ゴールを `lia`（線形整数算術の決定手続き）で**自動的に**閉じる．
- 補題 1 のフレーム部分は `Forall_map` ＋ `Forall_impl` で全要素へ一括適用．

■主定理の Coq 文（抜粋）.

```
Theorem type_safety :
  forall p r r' v,
    ptype p r r' -> has_rank r v ->
    exists vf, multistep r v p r' vf [] /\ has_rank r' vf.
Proof.
  induction p as [|o p IH]; intros r r' v Hty Hv.
  - inversion Hty; subst. exists v. split; [constructor | exact Hv].
  - inversion Hty as [| o0 p0 r0 r'0 Hle Hpt Heqp Heqr]; subst.
    assert (Hv2 : has_rank (r - rin o + rout o) (applyOp o (r - rin o) v)).
    { apply applyOp_rank. replace (r - rin o + rin o) with r by lia. exact Hv. }
    destruct (IH (r - rin o + rout o) r' (applyOp o (r - rin o) v) Hpt Hv2)
      as [vf [Hms Hvf]].
    exists vf. split.
    + eapply ms_step; [ apply st_op; exact Hle | exact Hms ].
```

```
    + exact Hvf.
Qed.
```

■非空虚性の確認. 抽象枠組みが充足可能であることを示すため, `Module Example` に具体例を実装した: スカラ = 自然数, 演算子 3 種 (`Oincr`: rank $0 \to 0$ で $+1$, `Osum`: rank $1 \to 0$ で総和, `Owrap`: rank $0 \to 1$ で長さ 1 のベクトルに包む). 唯一の義務 `mydenote_rank` は `destruct; inversion; repeat constructor` の 1 行で自動証明. パイプライン `[Owrap; Owrap; Osum]` が rank $0 \to 1$ に型付けされ, `Scalar 7` に対し実際に評価されてランク 1 の値になることを `type_safety` から導出した.

■コンパイル方法.

```
cd /Users/kodamay/projects/semantics
coqc AIPL.v      (* => 終了コード 0、AIPL.vo を生成（全証明が検査済み） *)
```

8 まとめ

- AIPL の公開実行モデル（ランク多相 = セル/フレーム）から，**型健全性が成り立つ核フラグメント**の構文を切り出した.
- 型 = ランクとする単純で**決定的な型推論**（`ptype`）を導入．静的条件 $\mathrm{rin}(o) \le r$ が stuck を防ぐ.
- **進行・保存・型安全性・決定性**を証明．well-typed なパイプラインは決して stuck せず，必ず予測ランクの値に有限ステップで評価される（停止性も付随）.
- すべてを **Coq (Rocq 9.1.0) で機械化し，自動的に検査されることを確認**した（`AIPL.v`）.

■次のステップ（論文入手後）. 論文本文を共有いただければ，(1) 半端ランク 0.5/1.5 や (2) 名前付き列のレコード型への拡張，(3) 論文の意味論との一致確認，を行える．証明骨格（進行 + 保存 + 補題 1）はそのまま再利用できる.