Actario: 定理証明支援系Coqによるアクターシステムの検証

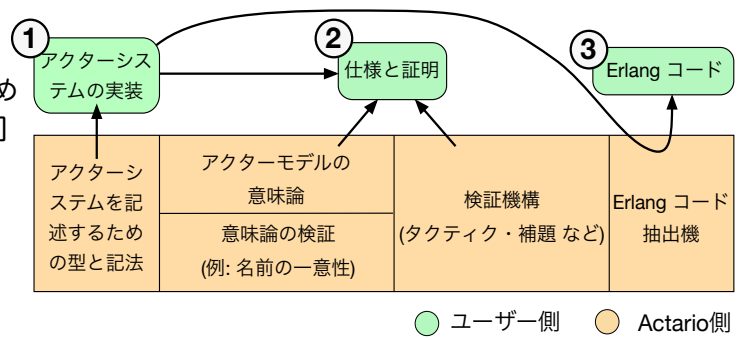
安武祥平・渡部卓雄 (東京工業大学)

背景と動機

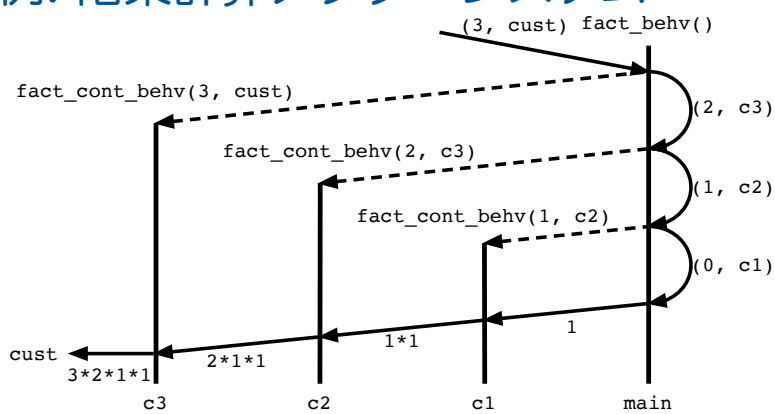
・アクターモデルを並行計算の基盤として持つ言語やライブラリ (例: Erlang, Akka) は現在広く使われている
・アクターモデルにより構築されたシステムの形式的な検証が課題となっている
・形式的な検証には主にモデル検査や定理証明によるアプローチがあるが、本研究では定理証明支援系を用いる

Actario https://github.com/amutake/actario

本研究で作成した、アクターシステムを記述・検証するためのCoqのライブラリ [Yasutake&Watanabe AGERE2015]
・アクターシステムを記述するための型と記法
・形式化されたアクターモデルの操作的意味論
・アクターシステムに関する命題を証明するための補題やタクティク
・Erlangへの抽出機構



例: 階乗計算アクターシステム



Actarioを使ったコード

このシステムに関する検証の例

```
Definition fact_cont_behv
    (state : ContState) :=
  receive (fun msg =>
    match msg, state with
    | nat_msg arg, val_cust val cust =>
      cust ! nat_msg (val * arg);
      become cont_done
    | _, _ => become state
    end).

Definition fact_behv (state : unit) :=
  receive (fun msg =>
    match msg with
    | tuple_msg (nat_msg (S n))
                (name_msg cust) =>
      cont <- new fact_cont_behv
              with (val_cust (S n) cust);
      me <- self;
      me ! tuple_msg (nat_msg n)
                     (name_msg cont);
      become tt
    | tuple_msg (nat_msg 0)
          (name_msg cust) =>
      cust ! nat_msg 1;
      become tt
    | _ => become tt
    end).
```

```
Theorem receive_3 :
  eventually_receive (factorial_system 3 cust)
                     cust (nat_msg 6).
Proof.
  unfold_ev eventually_receive=> p p0 is_path.
  step_until_stop is_path p0.
  found 22 p22 p23.
Qed.
```

アクターの名前付け

・アクターモデルの意味論では、アクターの名前は常に一意なものが生成されると仮定されていることが多い
  ・Coqのような厳密な形式化では問題になる
・Actarioではシステムが一意な名前を生成する (証明済み)

名前付けの方法

・親アクターの名前と、親が過去に生成したアクターの数のペアを名前とする
・親アクターの内部状態の情報のみで名前を生成できるので、システムの合成や分解を行っても問題はない

アクターシステムの証明の方針

・非決定的な遷移を1つずつ追っていく
・遷移前のシステムの状態から、そこから遷移しうる遷移後のシステムの状態のリストを計算することができるので、プログラマがシステムの状態を書き下さなくてよい

主なタクティク

・step: 遷移を一つ進める。複数の遷移が考えられる場合はその数だけサブゴールを作る
・found: 目的の状態になったときに証明を終わらせる