Australian National University

Register Machine and Unary Recursive Functions in HOL4

Presenter: Zhuo (Zoey) Chen
zhuo.chen1@anu.edu.au
Supervisor: Michael Norrish
michael.norrish@data61.csiro.au

HOL4

- Interactive Theorem Prover (ITP)
- Interactive Theorem Proving: formalisation of proofs on computers

HOL4

- Record: p1 = <| Person := "Mary"; Fruits:= ["Apple", "Orange"]; Age := 21 |>
- Set: {1,293,45}
- Function Composition: f o g x is the same as f(g(x))

HOL4

- npair ($\otimes$): npair 3 2 = $3 \otimes 2$ = tri (3 + 2) + 2

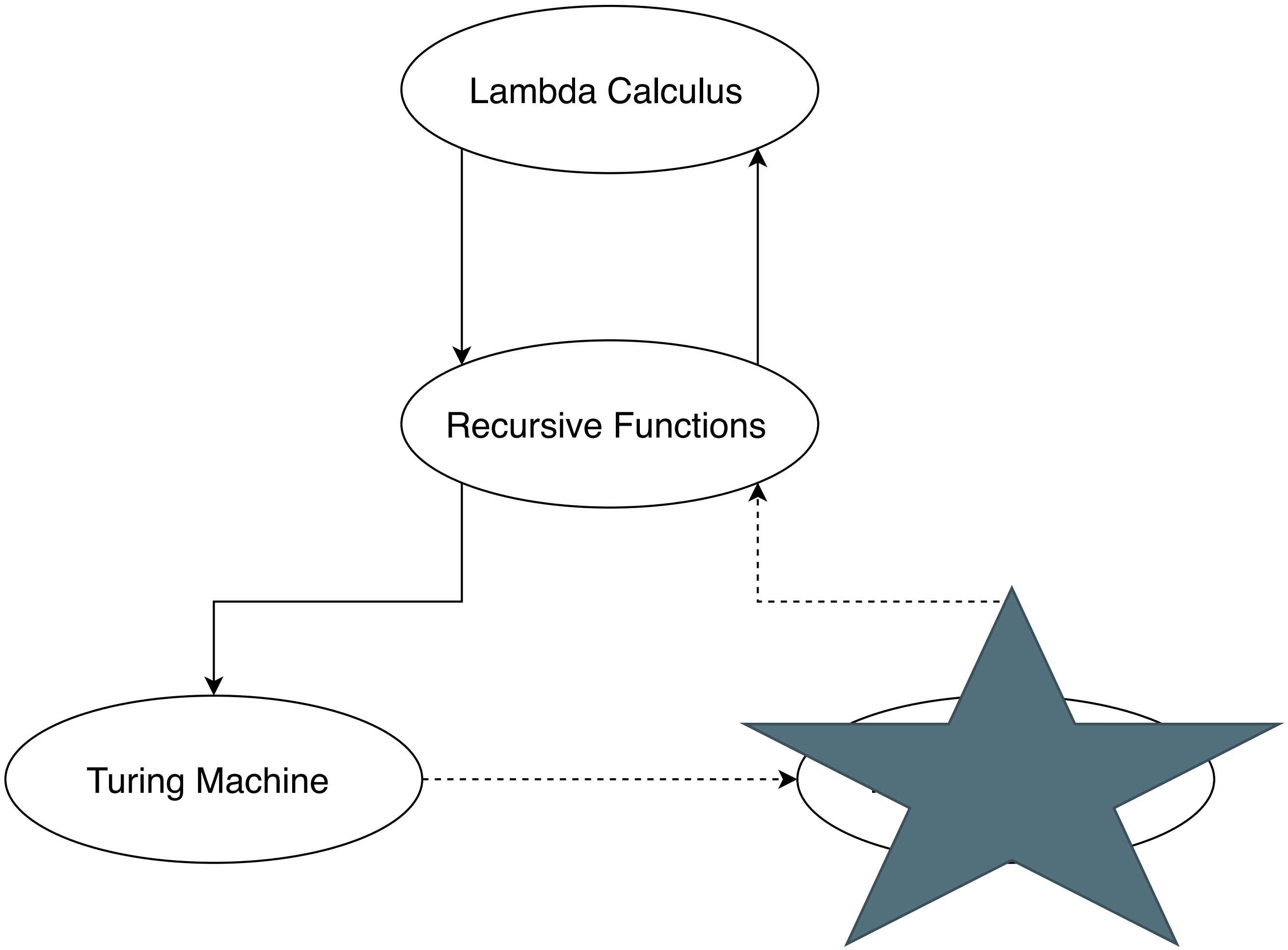
Let n = ($3 \otimes 2$)

- nfst ($3 \otimes 2$) = tri (tri^{-1} n) + tri^{-1} n − n =3
- nsnd ($3 \otimes 2$) = n − tri (tri^{-1} n) = 2

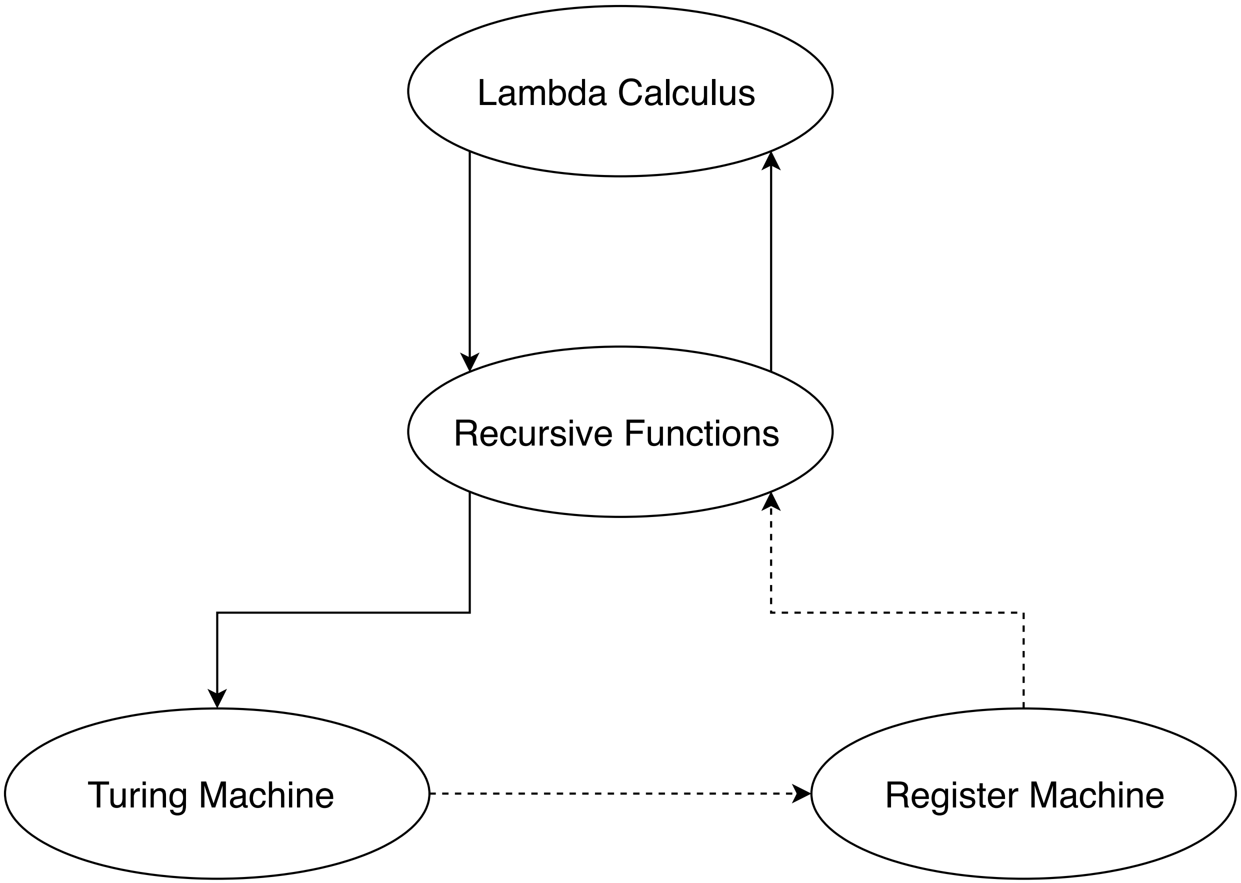
Equivalence

- Lambda Calculus
- Recursive Functions
- Turing Machines

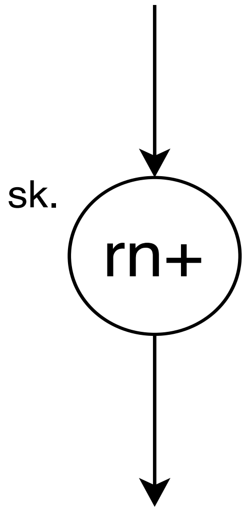
Equivalence



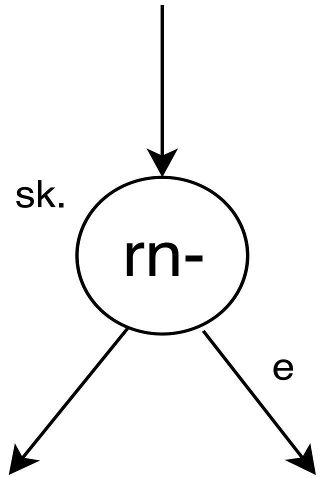
Equivalence



Register Machine



Add one stone to the basket n



Come out from arrow e if basket n is empty, otherwise remove one stone from basket n and come out from another arrow

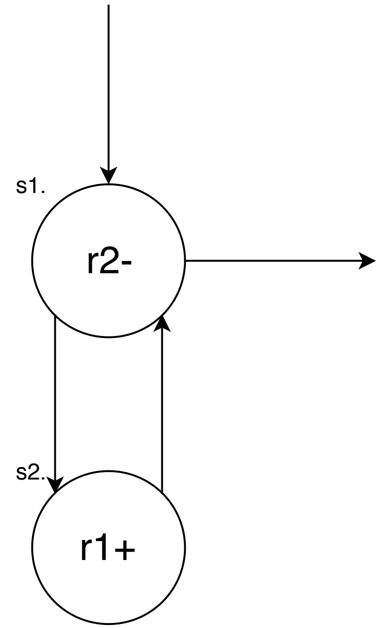
Register Machine in HOL4: Definition

rm=<|

Q : states
tf : num → action;

q0 : num;
In : num list;
Out : num

|>



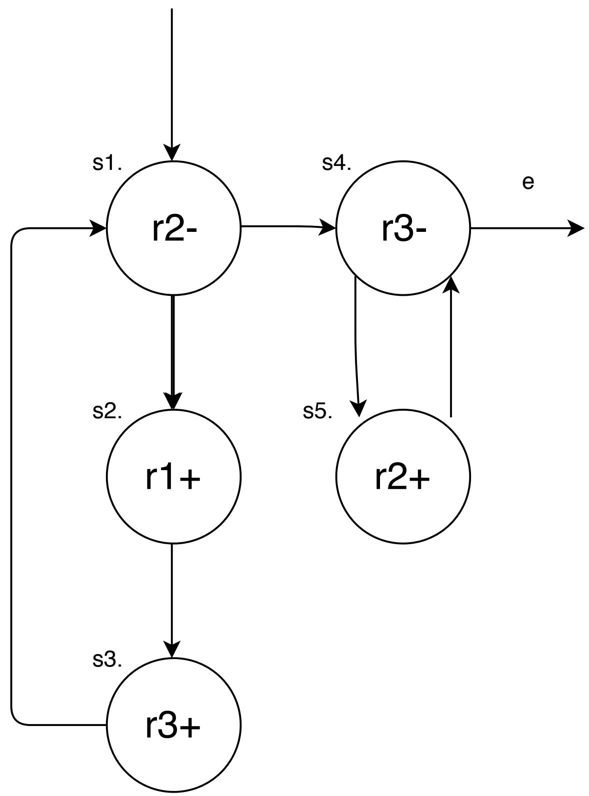
Wellformedness – wfrm

```
val wfrm_def = Define `
  wfrm m ⇔
    FINITE m.Q ∧
    m.q0 ∈ m.Q ∧
    (∀s. s ∈ m.Q ⇒ action_states (m.tf s) ⊆ m.Q)
`;
```

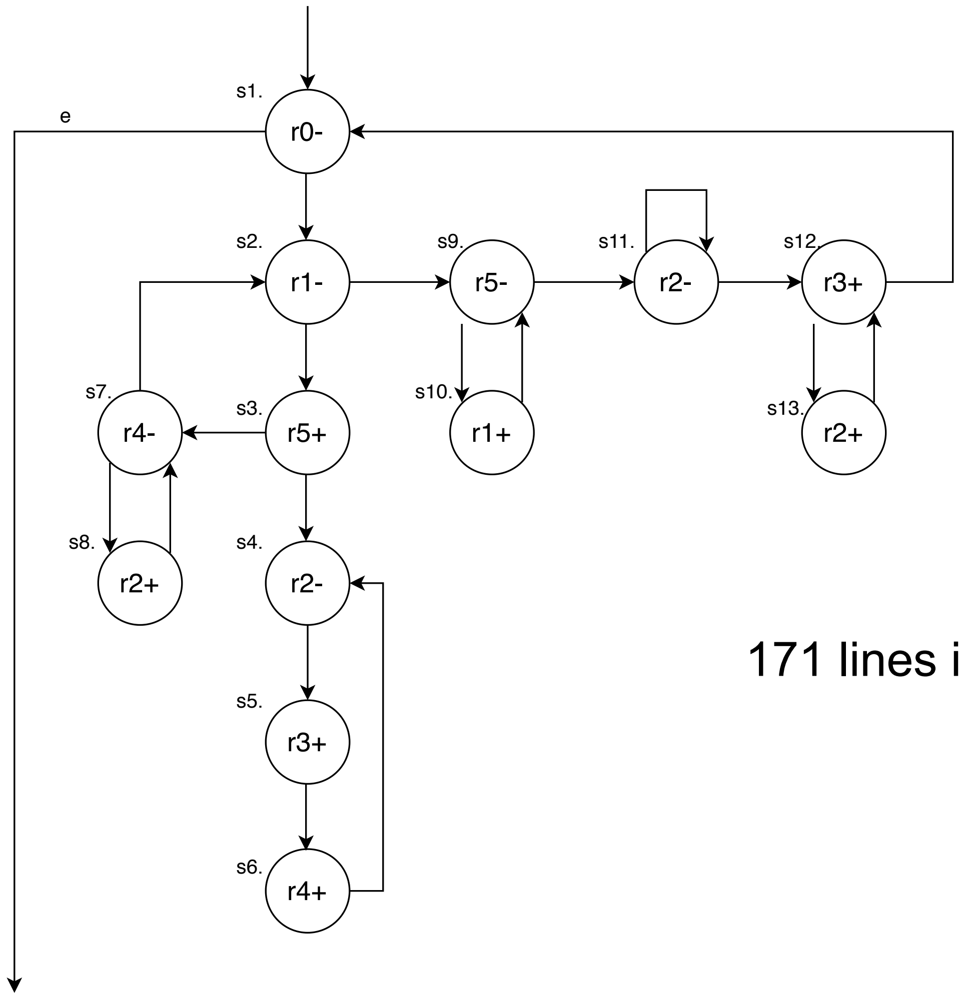
Register Machine in HOL4: Computation

- RUN
- Inputs initialisation + Check states and run

Register Machine in HOL4: Addition



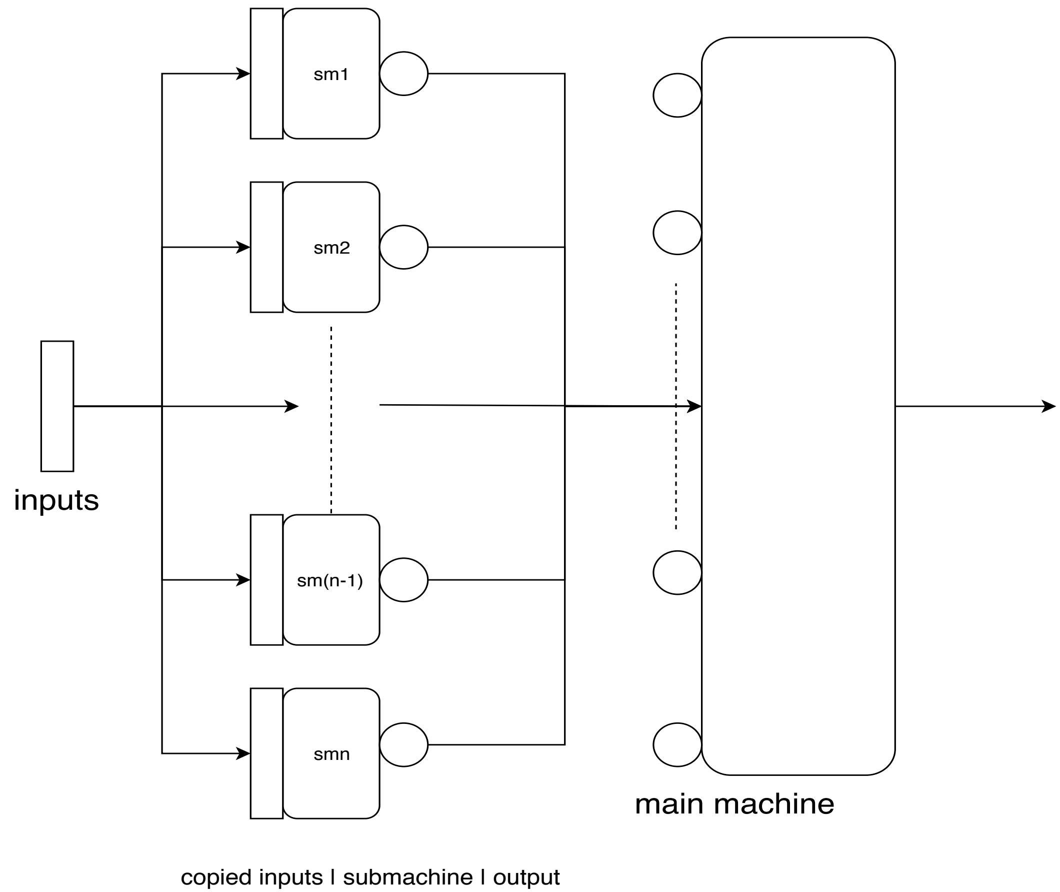
Register Machine in HOL4: Exponential



171 lines in total

Composition

Cn M mlist -> M o mlist



Better Verification Technology!

Verification Technology

- Hoare Triple and Lemmas
- Simple Sample Machines Proofs
- Glue Machines Proofs
- Correctness Definition

Verification Technology

- Hoare Triple and Lemmas
- Simple Sample Machines Proofs
- Glue Machines Proofs
- Correctness Definition

Hoare Triple

- {P} C {Q} (Classical Hoare Triple)
- {*P*, *q*0 } *M* {*Q*, *qf* } (Our Hoare Triple)

Description 1 *Hoare Triple Definition*

$$
\begin{array}{l}
\vdash \{P,q\}\ M\ \{Q,qf\} \iff \\
\quad \forall rs. \\
\quad\quad P\ rs \Rightarrow \\
\quad\quad \exists n\ rs'.\ \mathsf{run\_step}\ M\ (rs, \mathsf{SOME}\ q)\ n = (rs', qf) \land Q\ rs'
\end{array}
$$

Hoare Triple Lemmas

- *Sequential Composition*
- *Increment Correctness Preservation*
- *Decrement Correctness Preservation*
- *Lemma Weakening*
- *Loop Correctness*

Theorem 1 *Sequential Composition Hoare Triple Proof*

$$\vdash (\forall rs.\ Q\ rs \Rightarrow Q'\ rs) \wedge \{P, q_1\}\ m\ \{Q, \mathsf{SOME}\ q_2\} \wedge \\ \{Q', q_2\}\ m\ \{R, q_3\} \Rightarrow \\ \{P, q_1\}\ m\ \{R, q_3\}$$

Australian National University

Theorem 2 *Increment Correctness Preservation*

$$\vdash m.tf\, q_0 \;=\; \mathsf{Inc}\, r\, (\mathsf{SOME}\, d) \wedge q_0 \in m.Q \wedge$$
$$\{(\lambda\, rs.\, P\, rs(\!|r \mapsto rs\, r - 1|\!) \wedge 0 < rs\, r), d\}\, m\, \{Q, q\} \Rightarrow$$
$$\{P, q_0\}\, m\, \{Q, q\}$$

Theorem 3 *Decrement Correctness Preservation*

$$\begin{array}{l}\vdash m.\mathit{tf}\, q_0 \;=\; \mathsf{Dec}\; r\; (\mathsf{SOME}\; t)\; (\mathsf{SOME}\; e) \;\wedge\; q_0 \in m.Q \;\wedge \\ \quad \{(\lambda\, rs.\, P\; rs(\!|r \mapsto rs\; r + 1|\!)), t\}\; m\; \{Q, q\} \;\wedge \\ \quad \{(\lambda\, rs.\, P\; rs \wedge rs\; r = 0), e\}\; m\; \{Q, q\} \;\Rightarrow \\ \quad \{P, q_0\}\; m\; \{Q, q\}\end{array}$$

Theorem 4 *Lemma weakening*

$$\vdash (\forall s.\, P\, s \Rightarrow P'\, s) \wedge (\forall s.\, Q'\, s \Rightarrow Q\, s) \wedge \{P', q_0\}\, m\, \{Q', q\} \Rightarrow \{P, q_0\}\, m\, \{Q, q\}$$

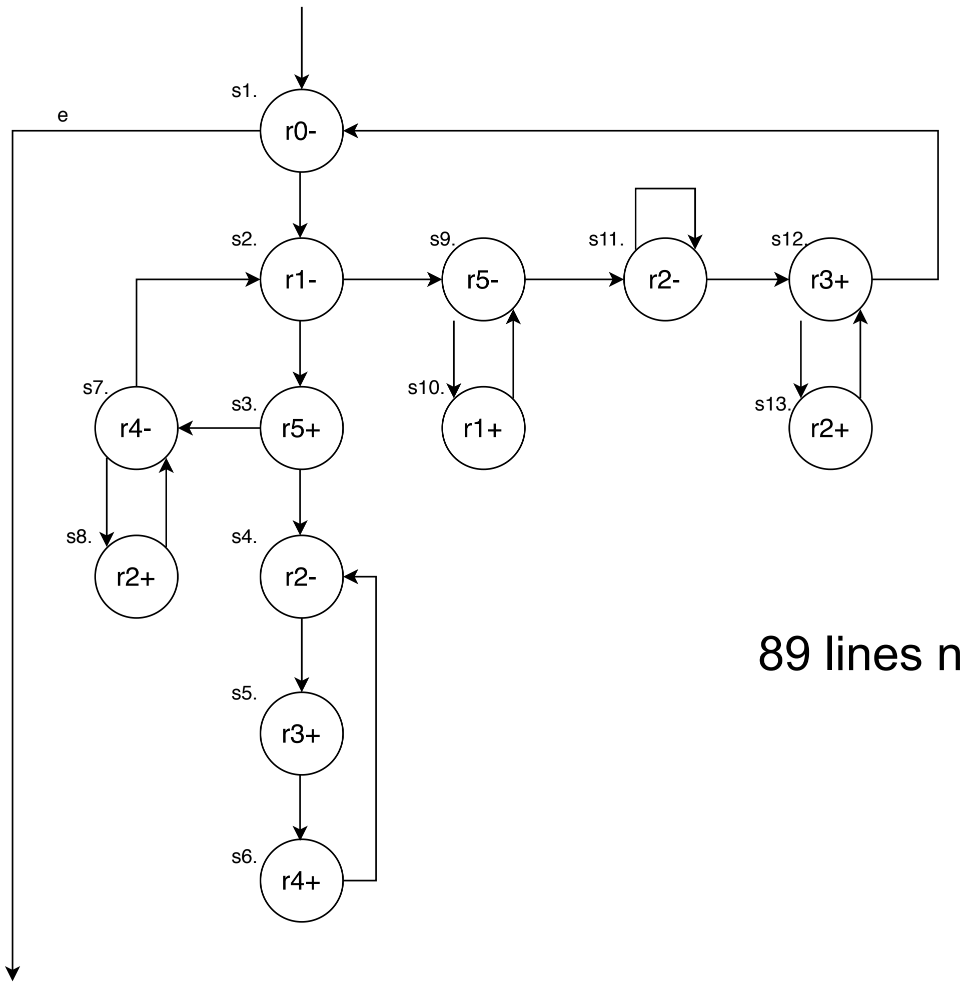
Theorem 5 *Loop Correctness*

$$
\begin{array}{l}
\vdash (\forall N. \\
\quad \{(\lambda\, rs. \\
\quad\quad INV\ rs(\!|gd \mapsto rs\ gd + 1|\!) \wedge \\
\quad\quad rs\ gd = N), body\}\ m \\
\quad \{(\lambda\, rs'.\, INV\ rs' \wedge rs'\ gd \leq N), \mathsf{SOME} \\
\quad\ q\}) \wedge (\forall rs.\, P\ rs \Rightarrow INV\ rs) \wedge \\
\ (\forall rs.\, INV\ rs \wedge rs\ gd = 0 \Rightarrow Q\ rs) \wedge \\
\ m.tf\ q = \mathsf{Dec}\ gd\ (\mathsf{SOME}\ body)\ exit \wedge \\
\ q \in m.Q \Rightarrow \\
\ \{P, q\}\ m\ \{Q, exit\}
\end{array}
$$

Verification Technology

- Hoare Triple and Lemmas
- Simple Sample Machines Proofs
- Glue Machines Proofs
- Correctness Definition

Exponential



89 lines now!

Verification Technology

- Hoare Triple and Lemmas
- Simple Sample Machines Proofs
- Glue Machines Proofs
- Correctness Definition

Glue Machines

Table 1: Glue Machines

Machine Name and Parameters	Description
dup r_1 r_2 r_3	Duplicates the value of r1 into r2 using r3 as scratch register
$m_1 \rightsquigarrow m_2$	Sequencial composition of m1 and m2 (link m2 onto the end of m1)
mrInst *mnum* *m*	Rename all the registers *r* used in machine *m* to $mnum \otimes r$
msInst *mnum* *m*	Rename all the states *s* in machine *m* to $mnum \otimes s$

Dup Hoare Triple Proof

- Starting state:

  rs r3 -> 0; rs r1 -> N; RS

- Ending state:

  r2 -> N; r3 -> 0; r1 -> N; Same as RS
except r2

Dup Hoare Triple Proof

- Starting state:

  rs r3 -> 0; rs r1 -> N; INV rs

- Ending state:

  r2 -> N; r3 -> 0; r1 -> N; INV rs

Theorem 9 *Duplication Hoare Triple Proof*

$$
\begin{array}{l}
\vdash r_1 \neq r_2 \wedge r_1 \neq r_3 \wedge r_2 \neq r_3 \wedge \\
\quad P = \\
\quad (\lambda\, rs. \\
\qquad rs\, r_3 = 0 \wedge rs\, r_1 = N \wedge \\
\qquad INV\ (\lambda\, r.\ \texttt{if}\ r \in \{ r_1; r_2; r_3 \}\ \texttt{then}\ 0\ \texttt{else}\ rs\, r)) \wedge \\
\quad Q = \\
\quad (\lambda\, rs. \\
\qquad rs\, r_2 = N \wedge rs\, r_1 = N \wedge rs\, r_3 = 0 \wedge \\
\qquad INV\ (\lambda\, r.\ \texttt{if}\ r \in \{ r_1; r_2; r_3 \}\ \texttt{then}\ 0\ \texttt{else}\ rs\, r)) \Rightarrow \\
\quad \{P, 0\}\ \mathsf{dup}\ r_1\ r_2\ r_3\ \{Q, \star\}
\end{array}
$$

Link Hoare Triple Proof

Theorem 10 *Link Hoare Triple Proof*

$$\begin{array}{l}\vdash \mathsf{wfrm}\ m_1\ \wedge\ \mathsf{wfrm}\ m_2\ \wedge\ \mathsf{DISJOINT}\ m_1.Q\ m_2.Q\ \wedge \\ q\ =\ m_1.q0 \wedge \{P, m_1.q0\}\ m_1\ \{Q, \star\}\ \wedge \\ \{Q', m_2.q0\}\ m_2\ \{R, opt\}\ \wedge\ (\forall rs.\ Q\ rs\ \Rightarrow\ Q'\ rs)\ \Rightarrow \\ \{P, q\}\ m_1\ \leadsto\ m_2\ \{R, opt\}\end{array}$$

- Used lemma: link_run_step

mrInst Hoare Triple Proof

Theorem 11 *Register Renaming Hoare Triple Proof*

$$
\begin{array}{l}
\vdash \mathsf{wfrm}\ M \land q \in M.Q \land \\
\quad P' = \\
\quad \mathsf{liftP\_V}\ mnum\ P \\
\quad\ (\lambda rs.\ \forall k.\ \mathsf{nfst}\ k \neq mnum \Rightarrow rs\ k = RS\ k) \land \\
\quad Q' = \\
\quad \mathsf{liftP\_V}\ mnum\ Q \\
\quad\ (\lambda rs.\ \forall k.\ \mathsf{nfst}\ k \neq mnum \Rightarrow rs\ k = RS\ k) \Rightarrow \\
\quad \{P, q\}\ M\ \{Q, opt\} \Rightarrow \\
\quad \{P', q\}\ \mathsf{mrInst}\ mnum\ M\ \{Q', opt\}
\end{array}
$$

- Works the same
- Doesn't touch any other registers

msInst Hoare Triple Proof

Theorem 12 *State Renaming Hoare Triple Proof*

$$\begin{array}{l} \vdash \mathsf{wfrm}\ M \ \wedge\ q\ \in\ M.Q\ \Rightarrow \\ \quad \{P,q\}\ M\ \{Q,opt\}\ \Rightarrow \\ \quad \{P, mnum\ \otimes\ q\}\ \mathsf{msInst}\ mnum\ M \\ \quad \{Q, \mathsf{npair\_opt}\ mnum\ opt\} \end{array}$$

Verification Technology

- Hoare Triple and Lemmas
- Simple Sample Machines Proofs
- Glue Machines Proofs
- Correctness Definition

Correctness Definition

Given: Machine M simulates function f

- M correct if it works the same as f
- Describe how M \`works\` using Hoare Triple

Description 2 *Correctness Definition*

$$
\begin{aligned}
&\vdash \mathsf{correct1\_rmcorr}\, f\, M \iff \\
&\quad \exists min. \\
&\qquad M.\mathsf{In} = [min] \land \mathsf{wfrm}\, M \land \\
&\qquad \forall inp. \\
&\qquad\quad \{(\lambda\, rs.\ rs\, min = inp \land \forall k.\ k \neq min \Rightarrow rs\, k = 0), M. \\
&\qquad\quad q0\}\ M \\
&\qquad\quad \{(\lambda\, rs.\ rs\, M.Out = f\ inp), \star\}
\end{aligned}
$$

Th

Pro

$\vdash$

pro

But wait..

- What were we trying to do again?

But wait..

- What was our aim again?
- Register Machines to Recursive Functions

RMs to Recursive Functions

- To Unary Recursive Functions
- Proof of 0, Successor and Composition
- Graph Model of Primitive Recursion

RMs to Recursive Functions

- To Unary Recursive Functions
- Proof of 0, Successor and Composition
- Graph Model of Primitive Recursion

Unary v.s. N-ary Recursive Functions

- Equivalent!

- If you are very keen..
- https://github.com/HOL-Theorem-Prover/HOL/commit/367b06f0495d0a24664523a9369f7ab126f538ac

Unary Recursive Functions in HOL4

- Pair f g n -> (f n) ⊗ (g n)
- First n -> nfst n
- Second n -> nsnd n

HOL4

- npair ($\otimes$): npair 3 2 = $3 \otimes 2$ = tri (3 + 2) + 2

Let n = $(3 \otimes 2)$

- nfst $(3 \otimes 2)$ = tri (tri^{-1} n) + tri^{-1} n − n =3
- nsnd $(3 \otimes 2)$ = n − tri (tri^{-1} n) = 2

Unary Recursive Functions in HOL4

- Pair f g n -> (f n) ⊗ (g n)
- First n -> nfst n
- Second n -> nsnd n

- Helper machines: Tri, invTri simulating tri and tri^{-1}.
- Constructed and proved

RMs to Recursive Functions

- To Unary Recursive Functions
- Proof of 0, Successor and Composition
- Graph Model of Primitive Recursion

0, Successor and Composition Proofs

- 0 n -> 0
- Successor n -> SUC n
- Composition f g -> f o g

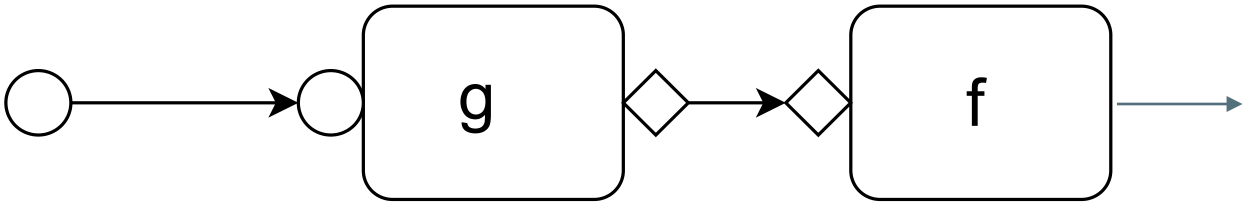
- 0 n -> 0

  const 0
- Successor n -> SUC n

  add1
- Composition f g -> f o g

  Cn

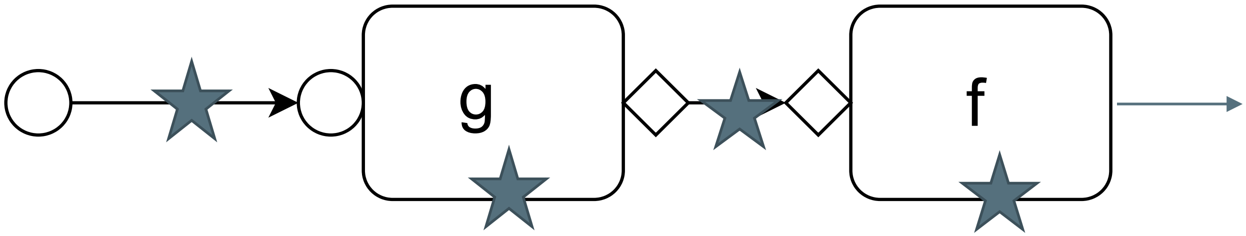
Composition: Definition



Composition

- Break into parts

Composition: Definition



- Requirement: No part changes anything outside their own registers (except dup)

- Requirement: No part changes anything outside their own registers (except dup)
- f: doesn't matter (last machine)

- Requirement: No part changes anything outside their own registers (except dup)
- f: doesn’t matter (last machine)
- dup : time to use the dup lemma!

Australian National University

Dup Hoare Triple Proof

- Starting state:

  rs r3 -> 0; rs r1 -> N; INV rs

- Ending state:

  r2 -> N; r3 -> 0; r1 -> N; INV rs

- Requirement: No part changes anything outside their own registers (except dup)
- f: doesn’t matter (last machine)
- dup : time to use the dup lemma!
- g: use the mrInst lemma!

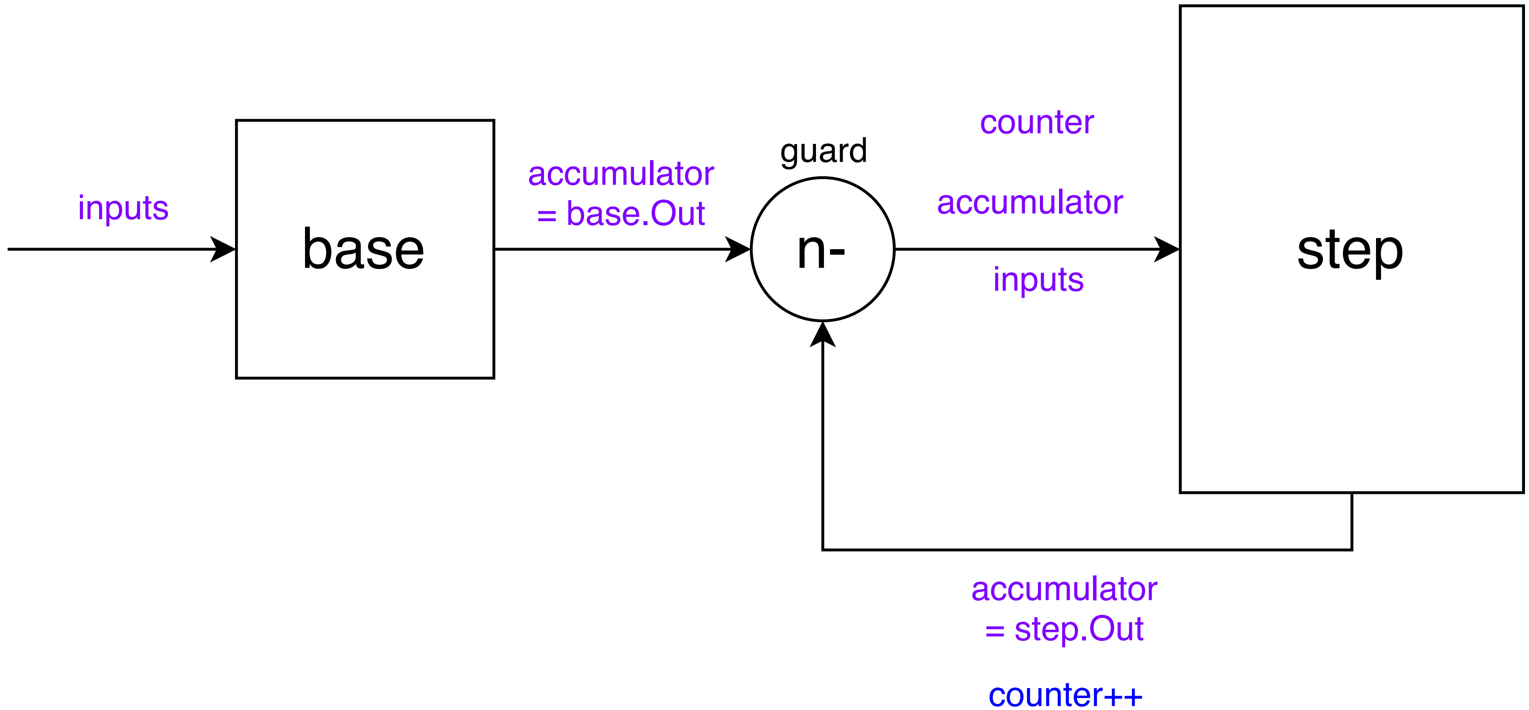
mrInst Hoare Triple Proof

- Works the same
- Doesn’t touch any other registers

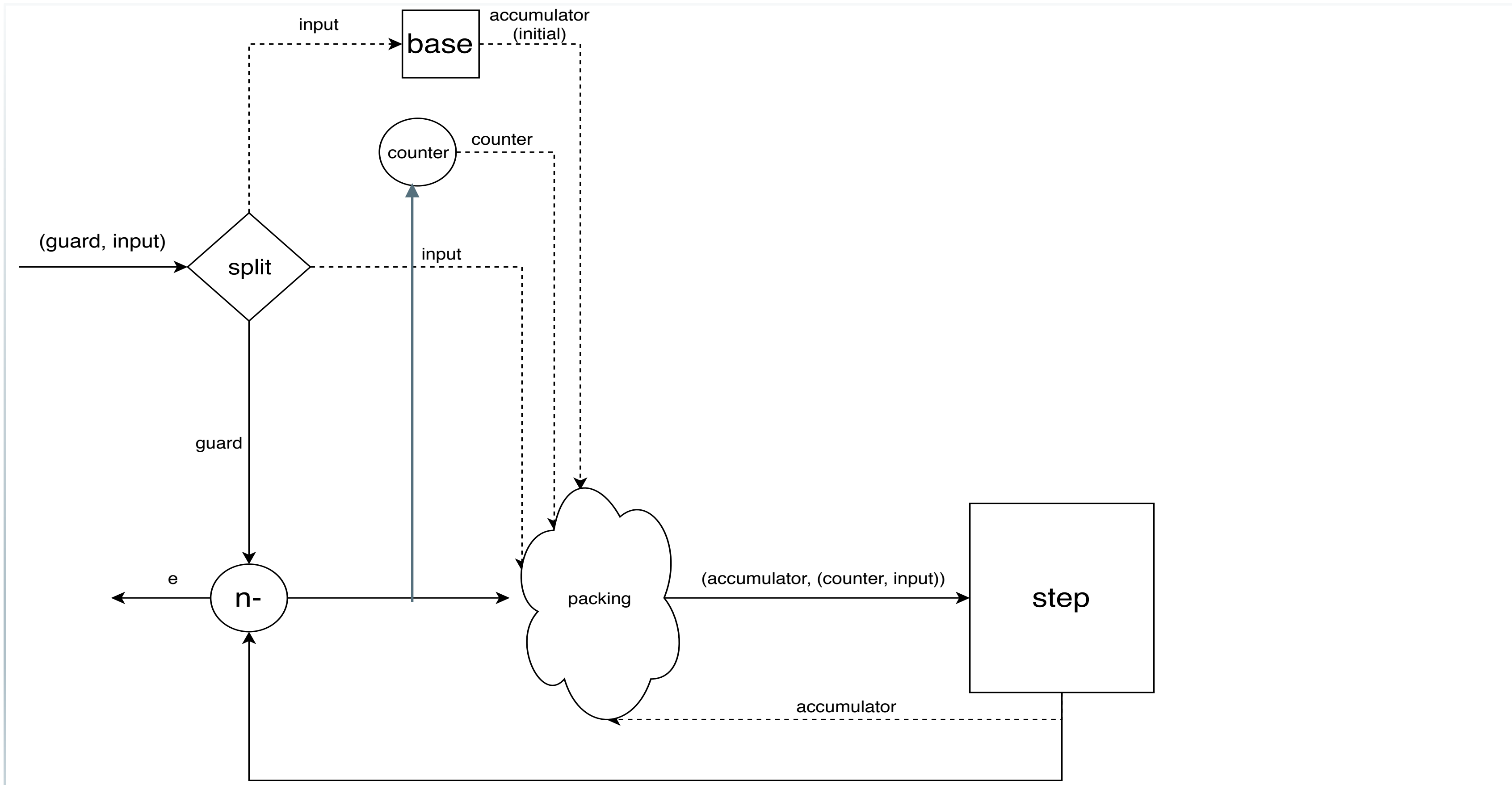
RMs to Recursive Functions

- To Unary Recursive Functions
- Proof of 0, Successor and Composition
- Graph Model of Primitive Recursion

N-ary Primitive Recursive Function Model



Unary Primitive Recursive Functions Model



Future Work

- Prove Pair, First, Second
- Unary Primitive Recursive Function
- Minimisation: Mu f n which finds the least n such that f(n) = 0
- Rewrite sample machines using Pr and Cn

Conclusion

- Hoare Triple and Correctness Definition
- Simple Sample Machines Proofs
- Glue Machines Proofs
- To Unary Recursive Functions
- Proof of 0, Successor and Composition
- Graph Model of Primitive Recursion
- Future Work

Australian National University

Global and Local Deducibility in Modal logic in HOL4

Presenter: Zhuo (Zoey) Chen
zhuo.chen1@anu.edu.au
Supervisor: Rajeev Gore
Rajeev.Gore@anu.edu.au

- Global: $\Gamma \vdash p$
- Local: $\Gamma \vdash (\ell)\, p$

Local Deducibility: Write $X \vdash_l \varphi$ iff there exists a finite subset $\{\psi_1, \psi_2, \cdots, \psi_n\} \subseteq X$ such that $\emptyset \vdash (\psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n) \rightarrow \varphi$

- Global: $\Gamma \vdash p$
- Local: $\Gamma \vdash (\ell)\ p$
- Difference:

$\varnothing \vdash (\ell)\ (A \text{ -> } B)$

$A \vdash (\ell)\ B$

$\varnothing \vdash (A \text{ -> } B)$

$A \vdash B$

Completeness Proof

- $\Gamma \vDash p$ implies $\Gamma \vdash P$
- Global deducibility
- Maximal consistent set -> local deducibility -> global decucibility -> MP

Completeness Proof

- $\Gamma \vDash p$ implies $\Gamma \vdash P$
- Global deducibility
- Maximal consistent set -> local deducibility -> global decucibility -> MP
- Global and local deducibility can simulate each other given Γ empty

Local Deducibility in HOL4

- By Yiming Xu
- Takes in Axioms
- Built-in rules

- K axiom: $\Box(A \to B) \to (\Box A \to \Box B)$
- Dual (instances): $\Diamond p \leftrightarrow \neg \Box \neg p$
- Propositional Tautology (instances)
- Modus Ponens
- Necessitation
- $\Diamond$ as primitive ($\Box = \neg\Diamond\neg$)

Global Deducibility in HOL4

- Id: $\Gamma \vdash p$ if p in Γ
- K Axiom: $\Box$(A->B) -> ($\Box$A -> $\Box$B)
- Propositional Tautology (instances)
- Modus Ponens
- Necessitation
- $\Diamond$ as primitive ($\Box = \neg\Diamond\neg$)

Problems encountered

- $\vdash \Diamond p \leftrightarrow \neg \Box \neg p$
- $\vdash \Diamond p \leftrightarrow \neg \neg \Diamond \neg \neg p$

Australian National University

Problems encountered

- $\vdash \Diamond p \leftrightarrow \neg \Box \neg p$
- $\vdash \Diamond p \leftrightarrow \neg \neg \Diamond \neg \neg p$

- $\vdash \Diamond p \leftrightarrow \Diamond \neg \neg p$
- $\vdash (\Diamond p \leftrightarrow \Diamond \neg \neg p) \leftrightarrow (\Diamond p \leftrightarrow \neg \neg \Diamond \neg \neg p)$
- MP

Australian National University

Problems encountered

- $\vdash \Diamond p \leftrightarrow \Diamond \neg \neg p$

- $\vdash$ A <-> B
- $\vdash$ X[N:=A] <-> X[N:=B]

Problems encountered

- $\vdash \Diamond p \leftrightarrow \Diamond \neg \neg p$

- $\vdash$ A <-> B
- $\vdash$ X[N:=A] <-> X[N:=B]

- Circled back to $\vdash \Diamond p \leftrightarrow \neg \Box \neg p$!

Australian National University

Global Deducibility in HOL4

- Id
- K axiom, Dual axiom
- Propositional Tautology (instances)
- MP
- Necessitation
- $\Diamond$ as primitive ($\Box = \neg\Diamond\neg$) on syntax level
- Instantiation achieved by substitution function

Local Deducibility <=> Global Deducibility

- Make $\vdash$ to take empty set of assumptions

Lemmas

- Substitution
- Propositional Tautology

Australian National University

```
Theorem gTk:
 ∀(p :num form list). (KGproof Ax p) ⇒ (∀f. (MEM f p) ⇒ gtt (Ax ∪ KDAxioms) ∅ f)
Proof
  Induct_on `KGproof` >> rw[] >> simp[gtt_rules, gttEmpG]
  >- metis_tac[gtt_rules]  (* MP *)
  >- (`subst (λs. if s = 0 then form1 else form2)
        (□ (VAR 0 -> VAR 1) -> □ (VAR 0) -> □ (VAR 1))
        = (□ (form1 -> form2) -> □ form1 -> □ form2) ` by simp[] >>
        `(□ (VAR 0 -> VAR 1) -> □ (VAR 0) -> □ (VAR 1)) ∈ (Ax ∪ KDAxioms)` by simp[KDAxioms_def] >>
      metis_tac[gtt_rules])  (* K axiom instance *)
(* DIAM q -> NOT BOX (NOT q) *)
  >- (`subst (λs. form) (DOUBLE_IMP (◇ (VAR 0)) (¬□ (¬VAR 0))) = (DOUBLE_IMP (◇ form) (¬□ (¬form)))`
         by simp[DOUBLE_IMP_def, IMP_def, BOX_def, subst_def, AND_def] >>
      `(DOUBLE_IMP (◇ (VAR 0)) (¬□ (¬VAR 0))) ∈ (Ax ∪ KDAxioms)`
         by simp[KDAxioms_def] >>
      `gtt (Ax ∪ KDAxioms) ∅ (DOUBLE_IMP (◇ form) (¬□ (¬form)))` by metis_tac[gtt_rules] >>
      irule gtt_double_imp_decompose_ltr >>
      rw[])
(* NOT BOX (NOT q) -> DIAM q *)
  >- (`subst (λs. form) (DOUBLE_IMP (◇ (VAR 0)) (¬□ (¬VAR 0))) = (DOUBLE_IMP (◇ form) (¬□ (¬form)))`
         by simp[DOUBLE_IMP_def, IMP_def, BOX_def, subst_def, AND_def] >>
      `(DOUBLE_IMP (◇ (VAR 0)) (¬□ (¬VAR 0))) ∈ (Ax ∪ KDAxioms)`
         by simp[KDAxioms_def] >>
      `gtt (Ax ∪ KDAxioms) ∅ (DOUBLE_IMP (◇ form) (¬□ (¬form)))` by metis_tac[gtt_rules] >>
      irule gtt_double_imp_decompose_rtl >>
      rw[])
  >- metis_tac[subst_ptaut, gtt_Ext, UNION_COMM, subst_self] (* ptaut f *)
  >> metis_tac[gtt_Ext, UNION_COMM, subst_self, gtt_rules]
QED
```

```
Theorem kTg:
  ∀Ax (f: num form). gtt (Ax ∪ KDAxioms) ∅ f ⇒ ∃ (p:num form list). MEM f p ∧ KGproof Ax p
Proof
  strip_tac >>
  `KGproof Ax []` by metis_tac[KGproof_rules] >>
  Induct_on `gtt` >> rw[]
  >- (qexists_tac `[f; subst s f]` >> rw[] >>
      `KGproof Ax [f]` by metis_tac[KGproof_rules, APPEND] >>
      metis_tac[KGproof_rules, APPEND, MEM])
  >- (fs[KDAxioms_def, CPLAxioms_def]
      >- (qexists_tac `[f; subst s f]` >> rw[] >>
      `KGproof Ax [f]` by metis_tac[KGproof_rules, APPEND] >>
      metis_tac[KGproof_rules, APPEND, MEM])
      >- (qexists_tac `[□ (s 0 -> s 1) -> □ (s 0) -> □ (s 1)]` >> rw[]
            >> metis_tac[KGproof_rules, APPEND, MEM])
      >> rw[DOUBLE_IMP_def]
      >> qabbrev_tac `A = IMP (DIAM (VAR 0)) (NOT (BOX (NOT (VAR 0))))`
      >> qabbrev_tac `B = IMP (NOT (BOX (NOT (VAR 0)))) (DIAM (VAR 0))`
      >> qexists_tac `[A; B;((VAR 0) -> ((VAR 1) -> (AND (VAR 0) (VAR 1))));
             A -> (B -> (AND A B)); B -> (AND A B); AND A B; subst s (AND A B)]`
      >> rw[]
      >> `subst (λs. if s = 0 then A else B) ( (VAR 0) -> ( (VAR 1) -> ( AND (VAR 0) (VAR 1) ) ) )
        = (A -> (B -> (AND A B)))` by simp[subst_def, IMP_def, AND_def]
      >> `KGproof Ax
          [A; B; VAR 0 -> VAR 1 -> AND (VAR 0) (VAR 1);
           subst (λs. if s = 0 then A else B) ( (VAR 0) -> ( (VAR 1) -> ( AND (VAR 0) (VAR 1) ) ) )]`
             by metis_tac[KGproof_rules, APPEND, ptaut_thms, MEM]
      >> rfs[]
      >> metis_tac[KGproof_rules, APPEND, MEM])
  >- (qexists_tac`p'++p++[f']` >> rw[] >>
      `KGproof Ax (p'++p)` by metis_tac[KGproof_APPEND] >>
      metis_tac[KGproof_rules, APPEND, MEM, KGproof_APPEND, MEM_APPEND])
  >> qexists_tac`p++[□ f]` >> rw[] >> metis_tac[KGproof_rules, APPEND, MEM, KGproof_APPEND, MEM_APPEND]
QED
```