NICTA

Tableaux for Verification of Data-Centric Processes

Andreas Bauer[1,2]
Peter Baumgartner[1,2]
Martin Diller[1]
Michael Norrish[1,2]

[1] NICTA
[2] ANU

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

Goal

Application viewpoint

To build a verification system for analysing temporal properties of **data-centric** (business) processes

Current technology is mainy Petri-Nets and propositional model checking

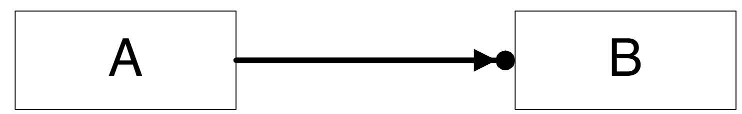
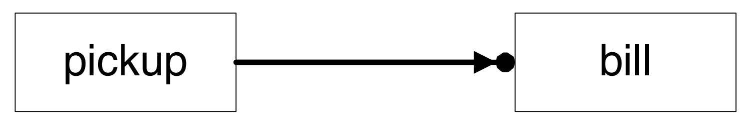
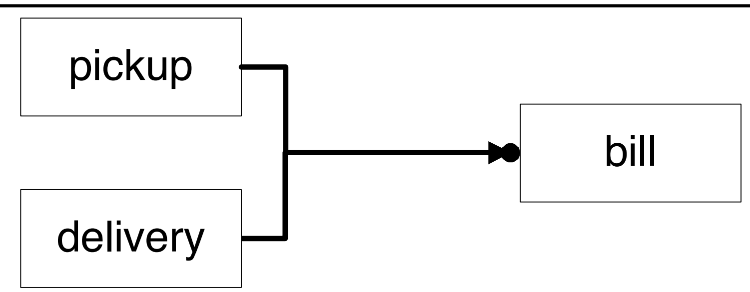
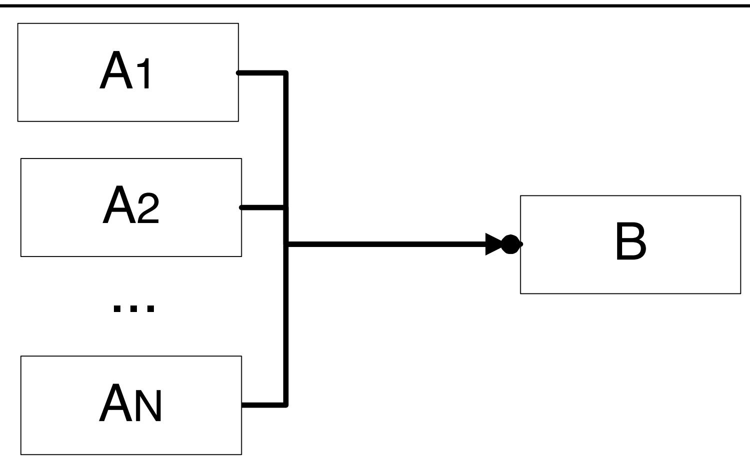
Tableaux viewpoint

To build a model checker for CTL*(FOL(Arrays+Lists+LIA))

Is it feasibly in practice despite (high) undecidability?

The Role of Propositional Model Checking

Modelling with process fragments in YAWL

	GRAPHICAL	LTL FORMULA
TEMPLATE		*(! A) W B*
'PLAIN' CONSTRAINT		*(! bill) W pickup*
BRANCHED CONSTRAINT		*(! bill) W (pickup V deliver)*
BRANCHED CONSTRAINT TO MULTIPLE TASKS		*(! B) W (A_1 V A_2 V … V A_N)*

We follow a similar approach but use FOL instead of PL

Talk Overview

1. Modelling Language and Reasoning Problems

2. Tableau calculus

3. Implementation and Experiments

Typed Data Modelling Language

NICTA

JSON Types

```
DB = {
  stock: Array[Stock],
  nrStockItems: Integer,
  open: List[Integer],
  gold: Boolean,
  invoice: Bool,
  paid: Bool,
  shipped: Bool }

Stock = {
  ident: String,
  price: Integer,
  available: Integer }
```

Typed Data Modelling Language

NICTA

JSON Types

```
DB = {
  stock: Array[Stock],
  nrStockItems: Integer,
  open: List[Integer],
  gold: Boolean,
  invoice: Bool,
  paid: Bool,
  shipped: Bool }

Stock = {
  ident: String,
  price: Integer,
  available: Integer }
```

Terms

(over FOL(Array+Records+List+LIA))

db.stock[head(*db*.open)].available - 1

db.open := tail(*db*.open)

Formulas

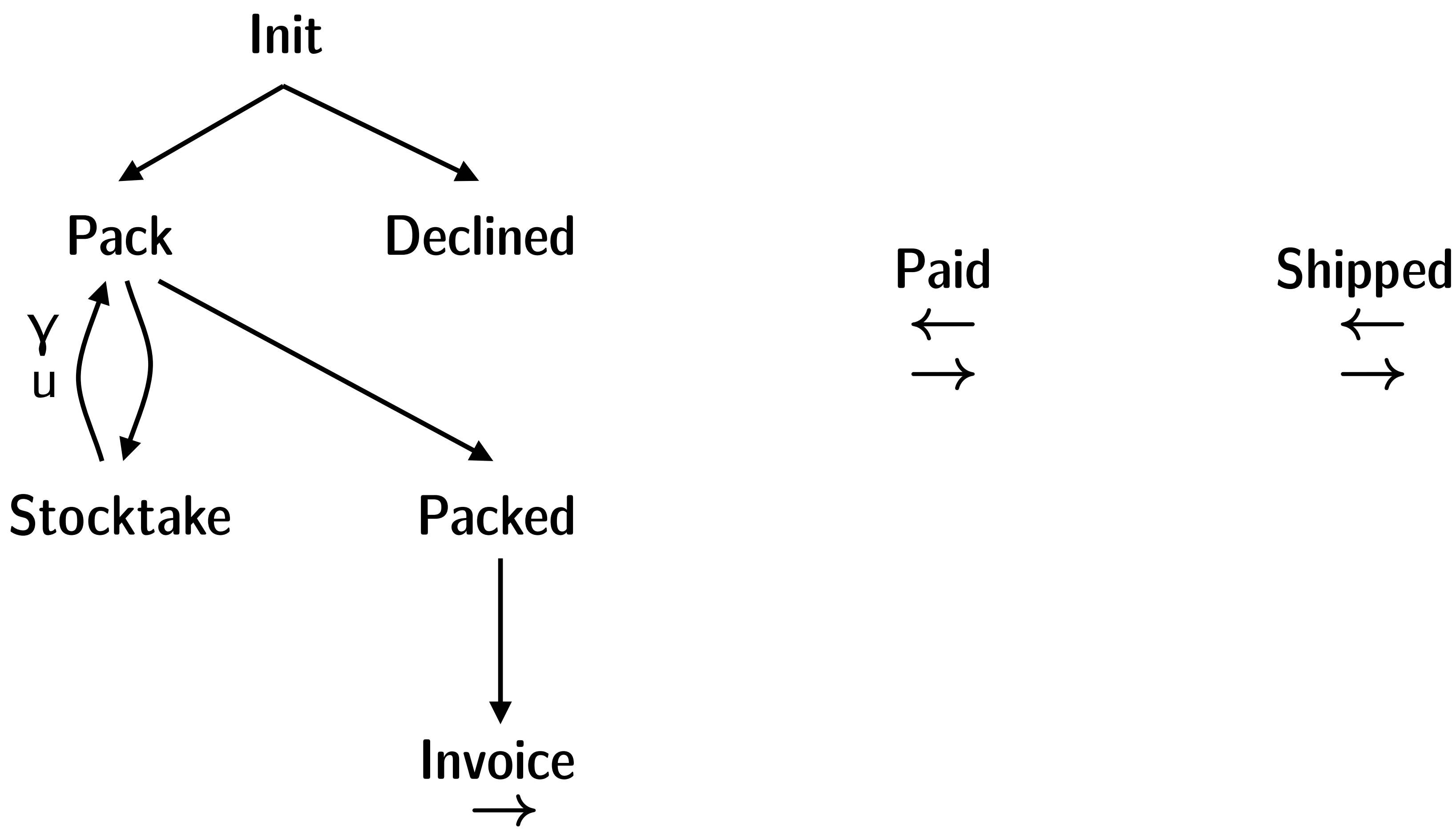
$\forall$ *db*:DB (acceptable(*db*) $\Leftrightarrow$ *db*.open $\neq$ nil)

Semantics

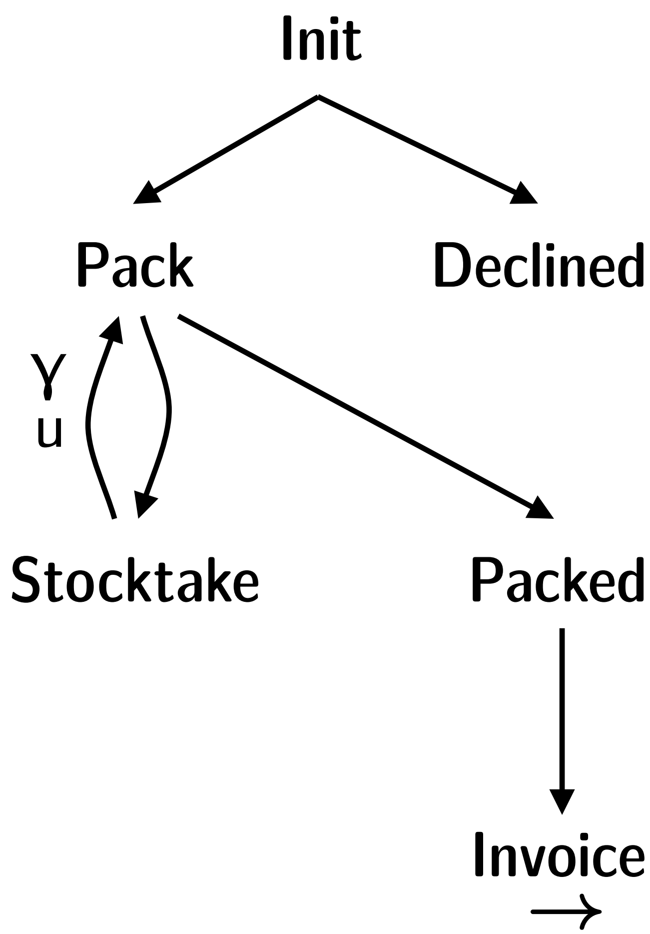
(I,α) $\models$ acceptable(*db*) $\wedge$ *db*.paid = false

where
I is an Array+Records+List+LIA interpretation and
α is an assignment to *db*

State Transition Systems (1)



State Transition Systems (1)



A state s is a pair (ℓ, α)

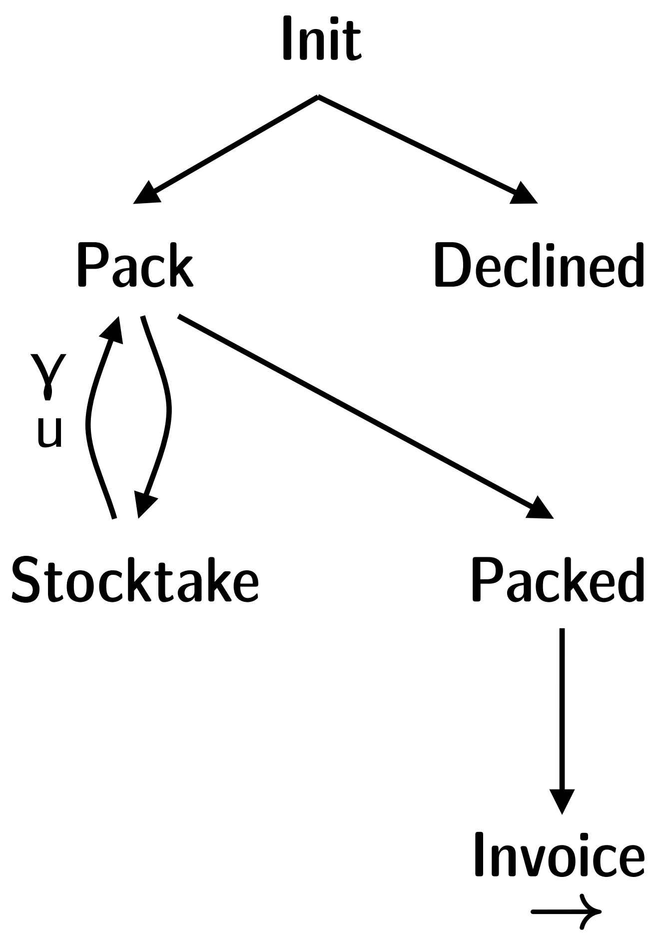
where ℓ is a **node** and

α is an assignment to db

Guard $\gamma[db]$

$db.\mathrm{stock}[\mathrm{head}(db.\mathrm{open})].\mathrm{available} > 0$

State Transition Systems (1)



A state s is a pair (ℓ , α)

where ℓ is a **node** and

α is an assignment to db

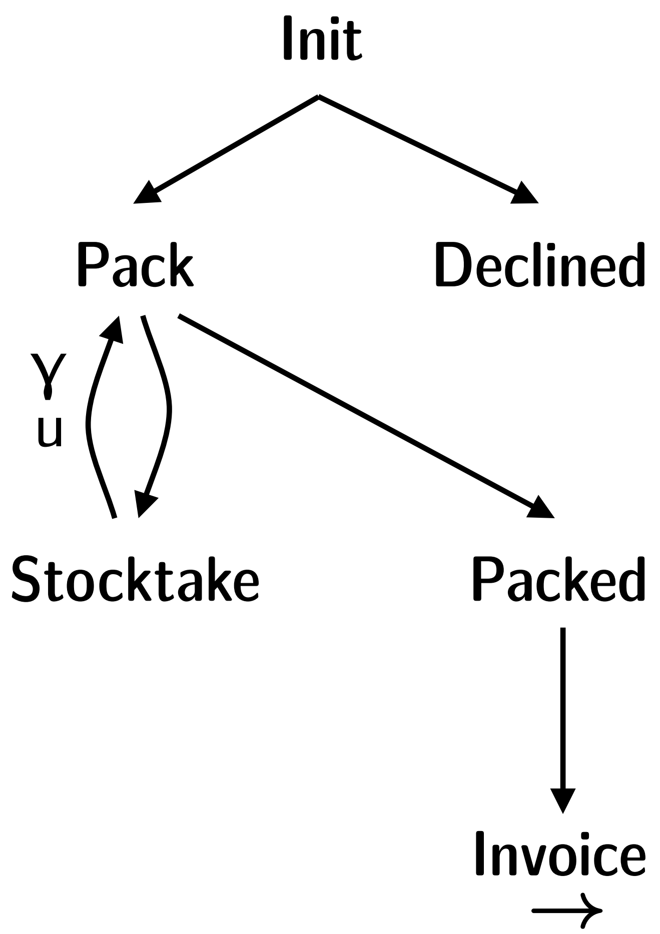
Guard γ[*db*]

db.stock[head(*db*.open)].available $>$ 0

Update term u[*db*]

db.stock[head(*db*.open)].available := *db*.stock[head(*db*.open)].available - 1;
db.open := tail(*db*.open)

State Transition Systems (2)



All fragment exit nodes $\rightarrow$ implicitly connected with all fragment entry nodes $\leftarrow$

CTL* constraints

$$db.\text{gold} = \text{false} \Rightarrow (db.\text{shipped} = \text{false}\ \mathbf{W}\ db.\text{paid} = \text{true}))$$

For non-gold customers no shipping until payment

Query Language CTL*

Syntax

State formulas $\Psi ::= \alpha[db] \mid \neg\Psi \mid \Psi \vee \Psi \mid \mathsf{E}\ \Phi \mid \mathsf{A}\ \Phi$

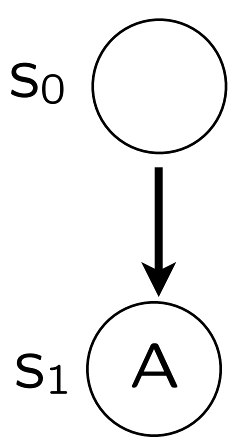
Path formulas $\Phi ::= \Psi \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathsf{X}\ \Phi \mid \mathsf{WX}\ \Phi \mid \Phi\ \mathsf{U}\ \Phi \mid \Phi\ \mathsf{R}\ \Phi$

First-order formulas $\alpha ::= \text{Atom} \mid \neg\alpha \mid \alpha \vee \alpha \mid \forall x\ \alpha$

(W, F, G, ∃ are macros)

Finite trace semantics [Manna&Pnueli 1995]

For e.g. X and WX



Query Language CTL*

NICTA

Syntax

State formulas $\Psi ::= \alpha[db] \mid \neg\Psi \mid \Psi \vee \Psi \mid \mathsf{E}\ \Phi \mid \mathsf{A}\ \Phi$

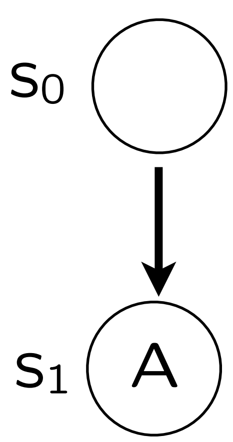
Path formulas $\Phi ::= \Psi \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathsf{X}\ \Phi \mid \mathsf{WX}\ \Phi \mid \Phi\ \mathsf{U}\ \Phi \mid \Phi\ \mathsf{R}\ \Phi$

First-order formulas $\alpha ::= \text{Atom} \mid \neg\alpha \mid \alpha \vee \alpha \mid \forall x\ \alpha$

(W, F, G, ∃ are macros)

Finite trace semantics [Manna&Pnueli 1995]

For e.g. X and WX



$s_0 s_1 \models \mathsf{X}\ \mathsf{A}$

Query Language CTL*

Syntax

State formulas $\Psi ::= \alpha[db] \mid \neg\Psi \mid \Psi \vee \Psi \mid \mathsf{E}\ \Phi \mid \mathsf{A}\ \Phi$

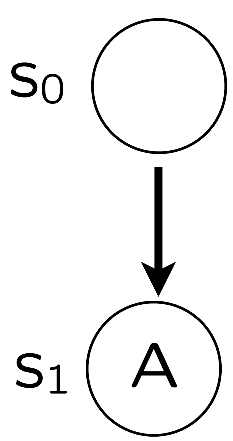
Path formulas $\Phi ::= \Psi \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathsf{X}\ \Phi \mid \mathsf{WX}\ \Phi \mid \Phi\ \mathsf{U}\ \Phi \mid \Phi\ \mathsf{R}\ \Phi$

First-order formulas $\alpha ::= \mathsf{Atom} \mid \neg\alpha \mid \alpha \vee \alpha \mid \forall x\ \alpha$

(W, F, G, ∃ are macros)

Finite trace semantics [Manna&Pnueli 1995]

For e.g. X and WX



$s_0 s_1 \models \mathsf{X}\ \mathsf{A}$ $s_0 s_1 \models \mathsf{WX}\ \mathsf{A}$

Query Language CTL*

Syntax

State formulas $\Psi ::= \alpha[db] \mid \neg\Psi \mid \Psi \vee \Psi \mid \mathrm{E}\ \Phi \mid \mathrm{A}\ \Phi$

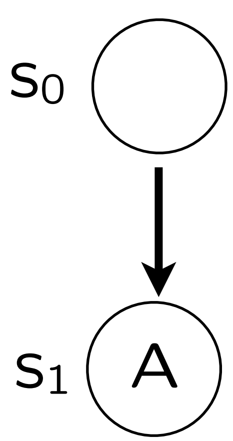
Path formulas $\Phi ::= \Psi \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathrm{X}\ \Phi \mid \mathrm{WX}\ \Phi \mid \Phi\ \mathrm{U}\ \Phi \mid \Phi\ \mathrm{R}\ \Phi$

First-order formulas $\alpha ::= \mathrm{Atom} \mid \neg\alpha \mid \alpha \vee \alpha \mid \forall x\ \alpha$

(W, F, G, $\exists$ are macros)

Finite trace semantics [Manna&Pnueli 1995]

For e.g. X and WX



$s_0 s_1 \models \mathrm{X}\ A$ $\qquad s_0 s_1 \models \mathrm{WX}\ A$ $\qquad s_0 s_1 \not\models \mathrm{X}\ B$

Query Language CTL*

NICTA

Syntax

State formulas $\Psi ::= \alpha[db] \mid \neg\Psi \mid \Psi \vee \Psi \mid \mathsf{E}\ \Phi \mid \mathsf{A}\ \Phi$

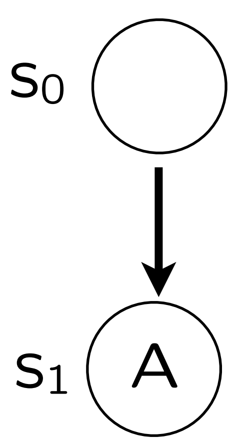
Path formulas $\Phi ::= \Psi \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathsf{X}\ \Phi \mid \mathsf{WX}\ \Phi \mid \Phi\ \mathsf{U}\ \Phi \mid \Phi\ \mathsf{R}\ \Phi$

First-order formulas $\alpha ::= \mathsf{Atom} \mid \neg\alpha \mid \alpha \vee \alpha \mid \forall x\ \alpha$

(W, F, G, $\exists$ are macros)

Finite trace semantics [Manna&Pnueli 1995]

For e.g. X and WX



$s_0 s_1 \models \mathsf{X}\ A$ $\quad s_0 s_1 \models \mathsf{WX}\ A$ $\quad s_0 s_1 \not\models \mathsf{X}\ B$ $\quad s_0 s_1 \not\models \mathsf{WX}\ B$

Query Language CTL*

NICTA

Syntax

State formulas $\Psi ::= \alpha[db] \mid \neg\Psi \mid \Psi \vee \Psi \mid \mathsf{E}\ \Phi \mid \mathsf{A}\ \Phi$

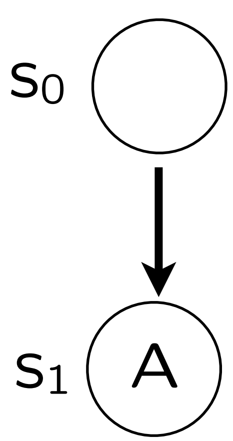
Path formulas $\Phi ::= \Psi \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathsf{X}\ \Phi \mid \mathsf{WX}\ \Phi \mid \Phi\ \mathsf{U}\ \Phi \mid \Phi\ \mathsf{R}\ \Phi$

First-order formulas $\alpha ::= \mathsf{Atom} \mid \neg\alpha \mid \alpha \vee \alpha \mid \forall x\ \alpha$

(W, F, G, $\exists$ are macros)

Finite trace semantics [Manna&Pnueli 1995]

For e.g. X and WX



$s_0 s_1 \models \mathsf{X}\ A$ $s_0 s_1 \models \mathsf{WX}\ A$ $s_0 s_1 \not\models \mathsf{X}\ B$ $s_0 s_1 \not\models \mathsf{WX}\ B$

$s_0 s_1 \models \mathsf{WX}\ \alpha$ for any α

Query Language CTL*

NICTA

Syntax

State formulas $\Psi ::= \alpha[db] \mid \neg\Psi \mid \Psi \vee \Psi \mid \mathsf{E}\ \Phi \mid \mathsf{A}\ \Phi$

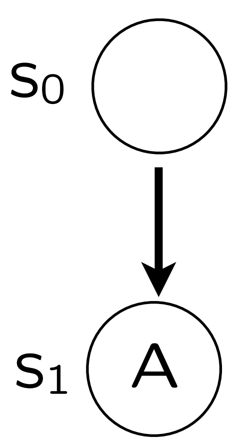
Path formulas $\Phi ::= \Psi \mid \neg\Phi \mid \Phi \vee \Phi \mid \mathsf{X}\ \Phi \mid \mathsf{WX}\ \Phi \mid \Phi\ \mathsf{U}\ \Phi \mid \Phi\ \mathsf{R}\ \Phi$

First-order formulas $\alpha ::= \mathsf{Atom} \mid \neg\alpha \mid \alpha \vee \alpha \mid \forall x\ \alpha$

(W, F, G, $\exists$ are macros)

Finite trace semantics [Manna&Pnueli 1995]

For e.g. X and WX

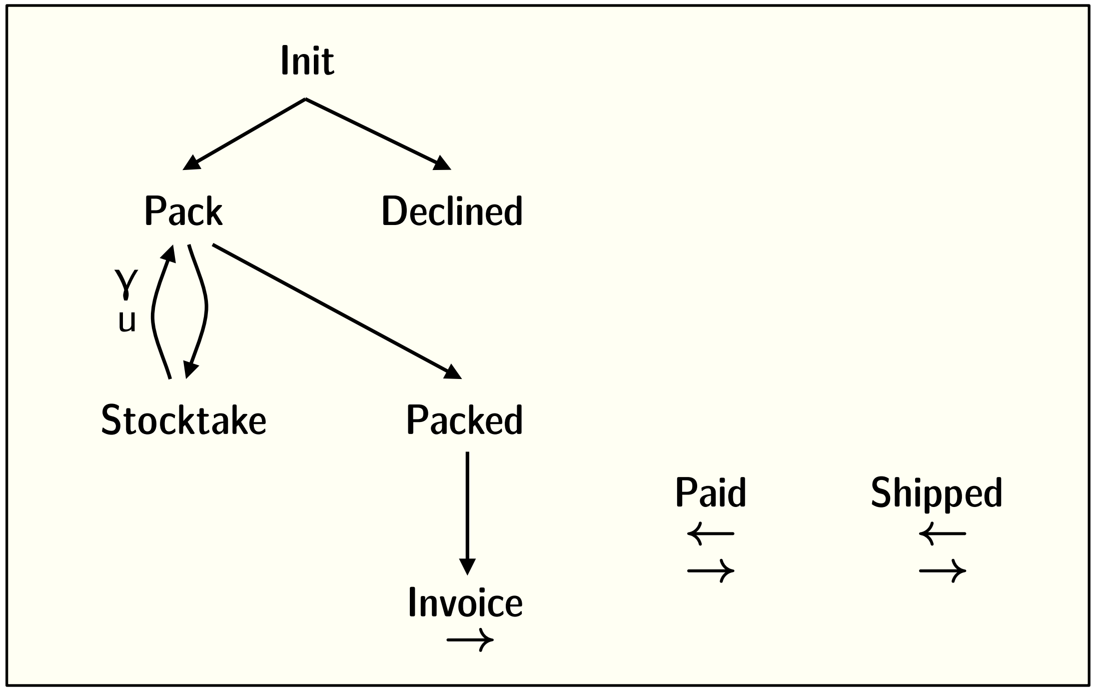


$s_0 s_1 \models \mathsf{X}\ A$ $\quad s_0 s_1 \models \mathsf{WX}\ A$ $\quad s_0 s_1 \not\models \mathsf{X}\ B$ $\quad s_0 s_1 \not\models \mathsf{WX}\ B$

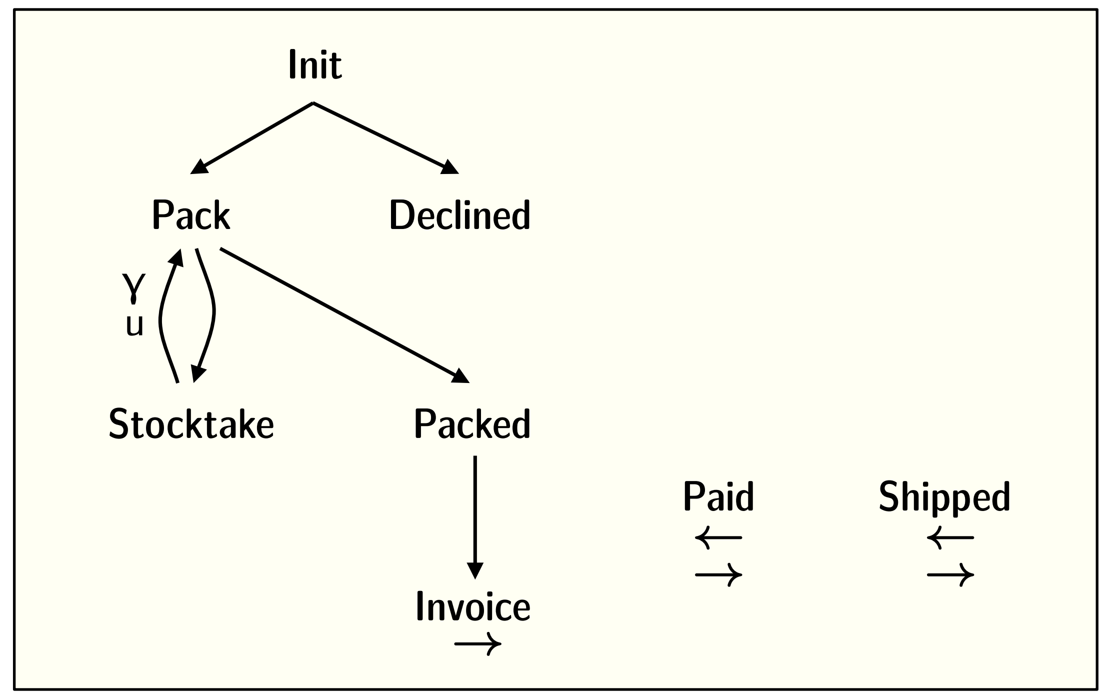
$s_0 s_1 \models \mathsf{WX}\ \alpha$ $\quad s_0 s_1 \not\models \mathsf{X}\ \alpha$ $\quad$ for any α

Satisfaction Relation

Satisfaction Relation



Satisfaction Relation

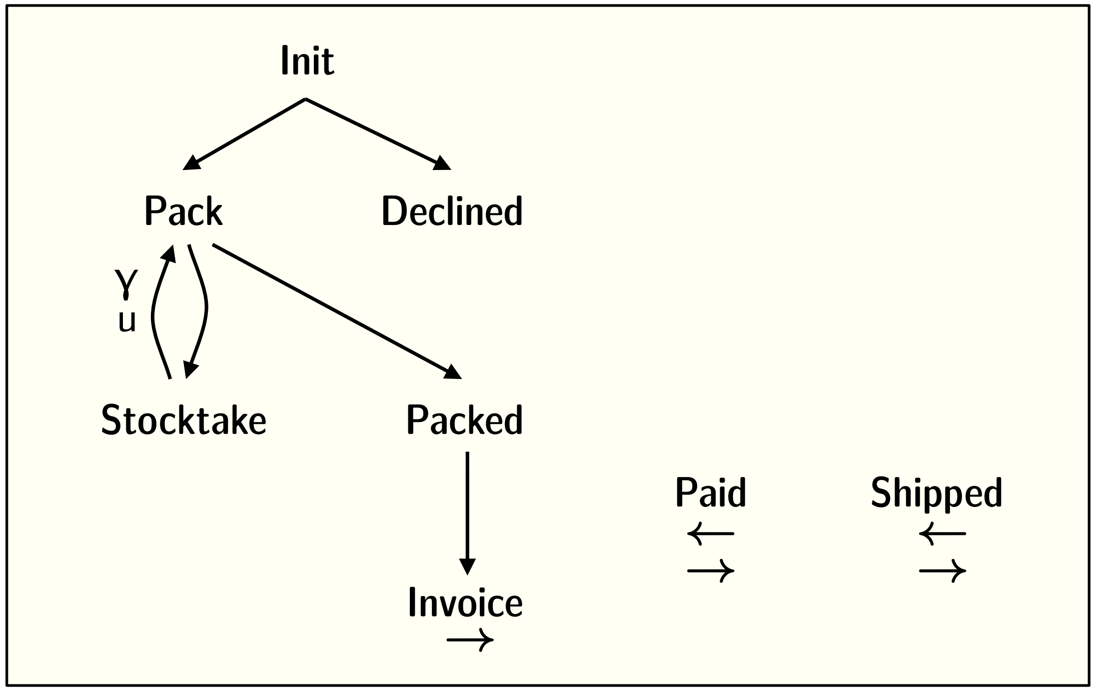


completed: $\forall$s:Status . (completed(s) $\Leftrightarrow$ (s.paid = true $\wedge$ s.shipped = true))
accepted: $\forall$db:DB . (acceptable(db) $\Leftrightarrow$ ($\neg$isEmpty(db.order)))
readyToShip: $\forall$s:Status . (readyToShip(s) $\Leftrightarrow$ (isEmpty(s.open))) ...

nongold: (db.gold = false $\Rightarrow$ (db.status.shipped = false W db.status.paid = true))

Satisfaction Relation

$\overset{?}{\models}$ Query



$\sqcap$

completed: $\forall$s:Status . (completed(s) $\Leftrightarrow$ (s.paid = true $\wedge$ s.shipped = true))
accepted: $\forall$db:DB . (acceptable(db) $\Leftrightarrow$ ($\neg$isEmpty(db.order)))
readyToShip: $\forall$s:Status . (readyToShip(s) $\Leftrightarrow$ (isEmpty(s.open))) ...

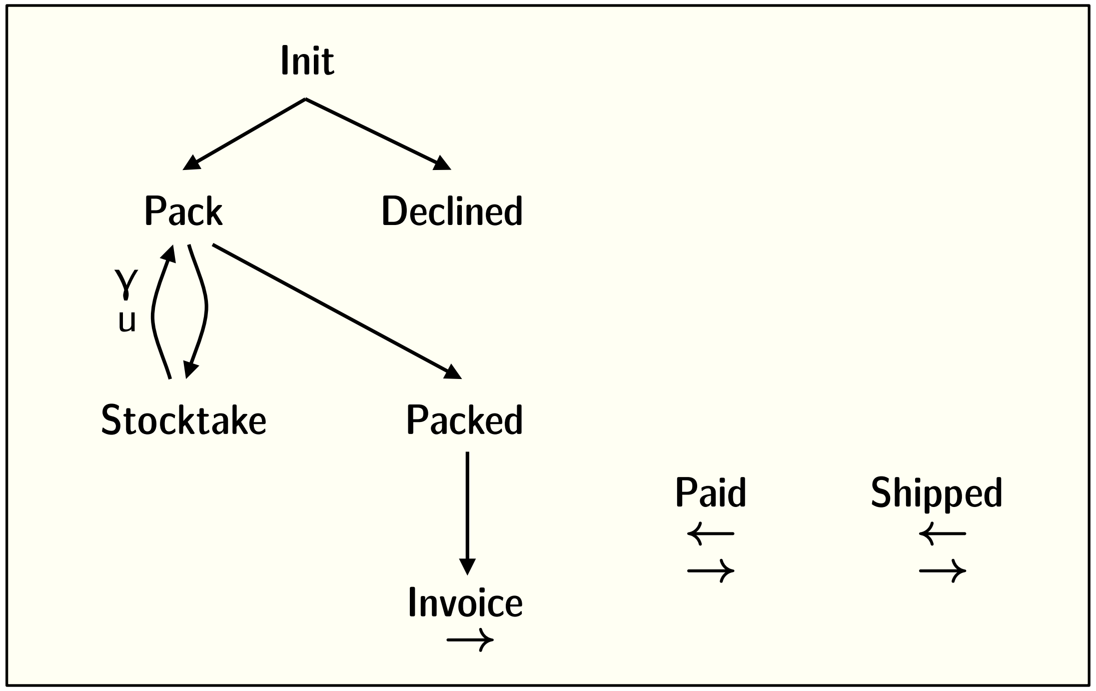
nongold: (db.gold = false $\Rightarrow$ (db.status.shipped = false W db.status.paid = true))

Satisfaction Relation

α

```
{   "order" : [1],
    "gold"  : true,
    "stock" : [ { "ident" : "Mouse",
                  "price" : 10,
                  "available" : 0  },
                { "ident" : "Monitor",
                  "price" : 200,
                  "available" : 2  },
                { "ident" : "Computer",
                  "price" : 1000,
                  "available" : 4  } ],
    "status" : {  "open" : [],
                  "value" : 0,
                  "shipping" : 0,
                  "paid" : false,
                  "shipped" : false,
                  "final" : false }    }
```

$\overset{?}{\models}$ Query



Π

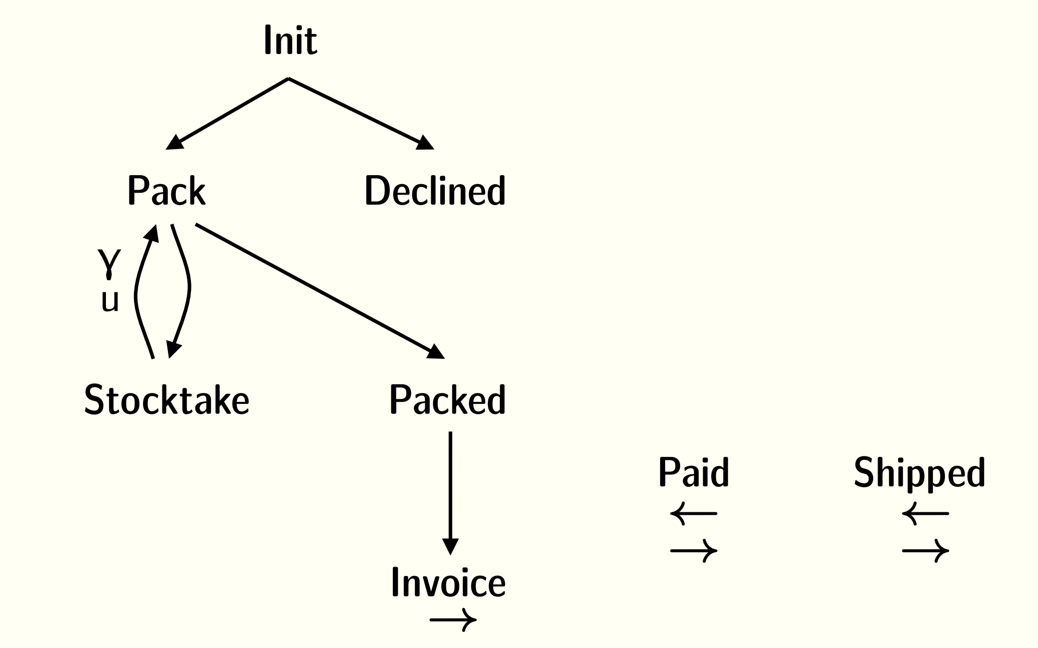
completed: $\forall$s:Status . (completed(s) $\Leftrightarrow$ (s.paid = true $\wedge$ s.shipped = true))
accepted: $\forall$db:DB . (acceptable(db) $\Leftrightarrow$ ($\neg$isEmpty(db.order)))
readyToShip: $\forall$s:Status . (readyToShip(s) $\Leftrightarrow$ (isEmpty(s.open))) ...

nongold: (db.gold = false $\Rightarrow$ (db.status.shipped = false W db.status.paid = true))

Satisfaction Relation

(α ,Init) $\overset{?}{\models}$ Query

```
{   "order" : [1],
    "gold"  : true,
    "stock" : [ { "ident" : "Mouse",
                  "price" : 10,
                  "available" : 0  },
                { "ident" : "Monitor",
                  "price" : 200,
                  "available" : 2  },
                { "ident" : "Computer",
                  "price" : 1000,
                  "available" : 4  } ],
    "status" : {  "open" : [],
                  "value" : 0,
                  "shipping" : 0,
                  "paid" : false,
                  "shipped" : false,
                  "final" : false }    }
```



⊓

completed: $\forall \mathtt{s:Status} \,.\, (\mathtt{completed(s)} \Leftrightarrow (\mathtt{s.paid} = \mathtt{true} \wedge \mathtt{s.shipped} = \mathtt{true}))$

accepted: $\forall \mathtt{db:DB} \,.\, (\mathtt{acceptable(db)} \Leftrightarrow (\neg \mathtt{isEmpty(db.order)}))$

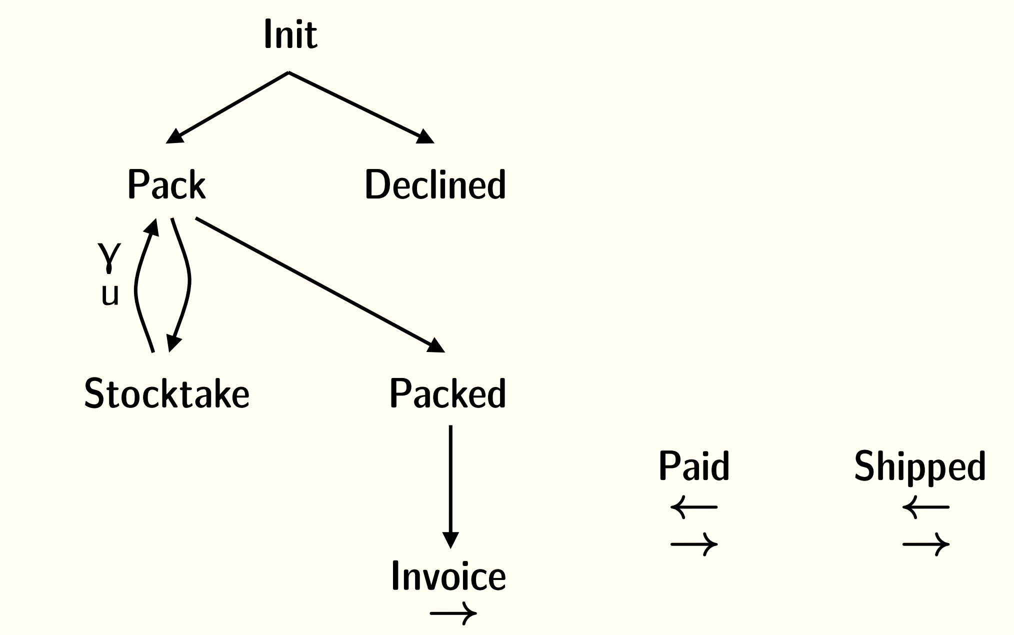
readyToShip: $\forall \mathtt{s:Status} \,.\, (\mathtt{readyToShip(s)} \Leftrightarrow (\mathtt{isEmpty(s.open)}))$ ...

nongold: $(\mathtt{db.gold} = \mathtt{false} \Rightarrow (\mathtt{db.status.shipped} = \mathtt{false} \; \mathsf{W} \; \mathtt{db.status.paid} = \mathtt{true}))$

Satisfaction Relation

$((\alpha, \text{Init}), I) \overset{?}{\models} \text{Query}$

```
{   "order" : [1],
    "gold"  : true,
    "stock" : [ { "ident" : "Mouse",
                  "price" : 10,
                  "available" : 0  },
                { "ident" : "Monitor",
                  "price" : 200,
                  "available" : 2  },
                { "ident" : "Computer",
                  "price" : 1000,
                  "available" : 4  } ],
    "status" : {  "open" : [],
                  "value" : 0,
                  "shipping" : 0,
                  "paid" : false,
                  "shipped" : false,
                  "final" : false }    }
```



Π

completed: $\forall$s:Status . (completed(s) $\Leftrightarrow$ (s.paid = true $\wedge$ s.shipped = true))
accepted: $\forall$db:DB . (acceptable(db) $\Leftrightarrow$ ($\neg$isEmpty(db.order)))
readyToShip: $\forall$s:Status . (readyToShip(s) $\Leftrightarrow$ (isEmpty(s.open))) ...

nongold: (db.gold = false $\Rightarrow$ (db.status.shipped = false W db.status.paid = true))

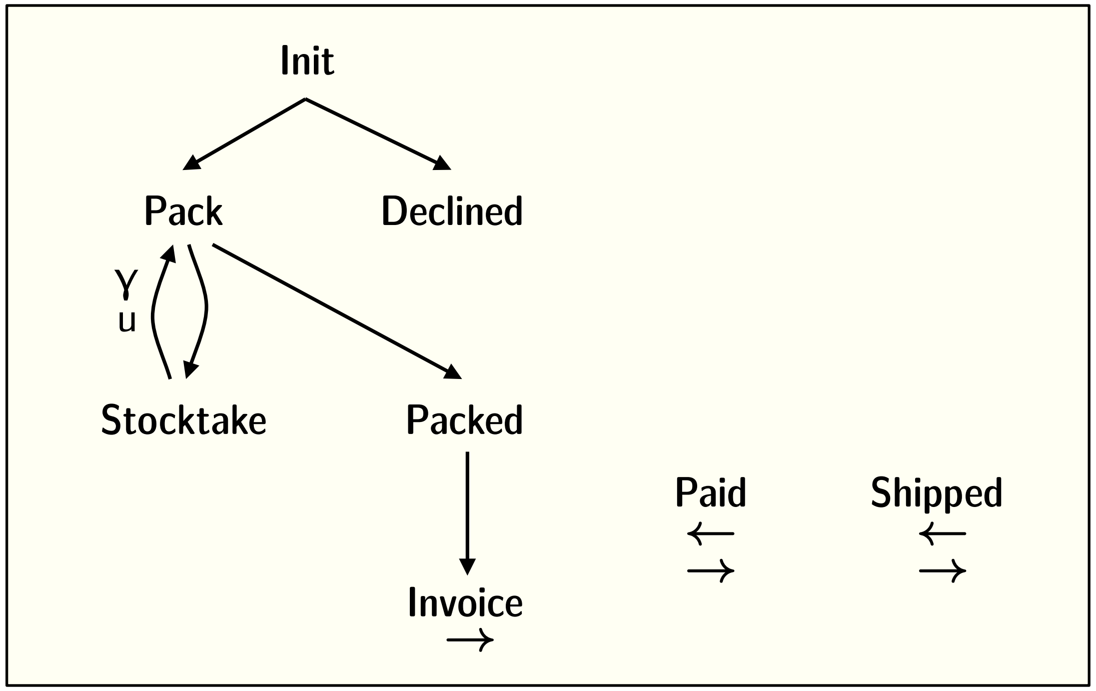
Satisfaction Relation

$$((\alpha \quad , \mathit{Init}) , I) \overset{?}{\models} E\,(\Pi \wedge F\ \mathit{db}.\mathrm{status.final}=\mathrm{true})$$

```
{   "order" : [1],
    "gold"  : true,
    "stock" : [ { "ident" : "Mouse",
                  "price" : 10,
                  "available" : 0  },
                { "ident" : "Monitor",
                  "price" : 200,
                  "available" : 2  },
                { "ident" : "Computer",
                  "price" : 1000,
                  "available" : 4  } ],
    "status" : {  "open" : [],
                  "value" : 0,
                  "shipping" : 0,
                  "paid" : false,
                  "shipped" : false,
                  "final" : false }    }
```

Is some final state reachable?

Planning task



Π

completed: $\forall s{:}Status\ .\ (completed(s) \Leftrightarrow (s.paid = true \wedge s.shipped = true))$
accepted: $\forall db{:}DB\ .\ (acceptable(db) \Leftrightarrow (\neg isEmpty(db.order)))$
readyToShip: $\forall s{:}Status\ .\ (readyToShip(s) \Leftrightarrow (isEmpty(s.open)))\ \ldots$

nongold: $(db.gold = false \Rightarrow (db.status.shipped = false\ W\ db.status.paid = true))$

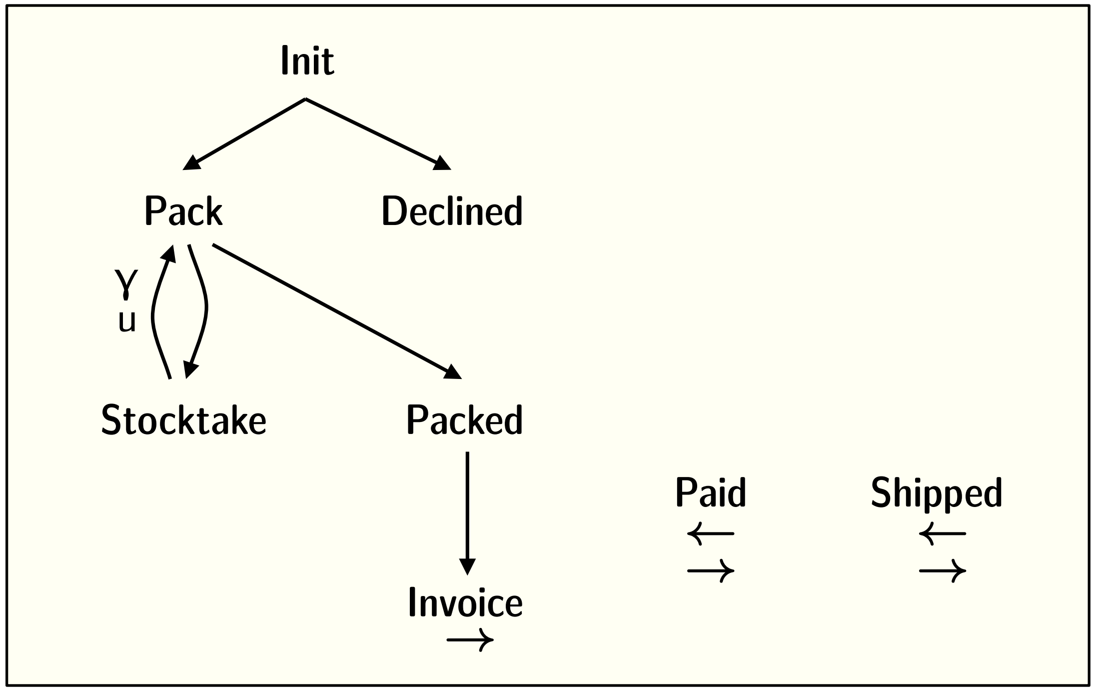
General Model Checking Problem

((α ,Init) ,I) $\models$ E (Π ∧ F *db*.status.final=true)

? ?

Is some final state reachable?

Planning task



Π

completed: ∀s:Status . (completed(s) ⇔ (s.paid = true ∧ s.shipped = true))
accepted: ∀db:DB . (acceptable(db) ⇔ (¬isEmpty(db.order)))
readyToShip: ∀s:Status . (readyToShip(s) ⇔ (isEmpty(s.open))) . . .

nongold: (db.gold = false ⇒ (db.status.shipped = false W db.status.paid = true))

Tableau Calculus

Essentially

Symbolic execution of the state transition system

Reduction to pure FOL proof problems

Unsatisfiability of the FOL proof problems proves the given (temporal) query unsatisfiable

Main data structure: Sequent $(\mathbf{m}, \mathbf{t}) \vdash_{\mathbf{Q}} \phi_1, \ldots, \phi_n$

m: node name, the current node

t: a ground term, the current database

$\mathbf{Q} \in \{ \mathbf{E}, \mathbf{A} \}$ path quantifier context

$\phi_i[db]$: formulas; read conjunctively if $\mathbf{Q} = \mathbf{E}$, disjunctively if $\mathbf{Q} = \mathbf{A}$

Tableau nodes are conjunctions of sequents

Tableau branches out disjunctively

Tableau Calculus

NICTA

Boolean Rules *where* $s = (\mathsf{m}, \mathsf{t})$

$$\text{E-}\wedge \ \frac{s \vdash_{\mathsf{E}} \phi \wedge \psi, \Phi; \Sigma}{s \vdash_{\mathsf{E}} \phi, \psi, \Phi; \Sigma} \qquad \text{E-}\vee \ \frac{s \vdash_{\mathsf{E}} \phi \vee \psi, \Phi; \Sigma}{s \vdash_{\mathsf{E}} \phi, \Phi; \Sigma \quad s \vdash_{\mathsf{E}} \psi, \Phi; \Sigma}$$

$$\text{A-}\vee \ \frac{s \vdash_{\mathsf{A}} \phi \vee \psi, \Phi; \Sigma}{s \vdash_{\mathsf{A}} \phi, \psi, \Phi; \Sigma} \qquad \text{A-}\wedge \ \frac{s \vdash_{\mathsf{A}} \phi \wedge \psi, \Phi; \Sigma}{s \vdash_{\mathsf{A}} \phi, \Phi; s \vdash_{\mathsf{A}} \psi, \Phi; \Sigma}$$

Rules for Path Quantifiers

$$\text{E-Elim} \ \frac{s \vdash_{\mathsf{E}} Q\,\phi, \Phi; \Sigma}{s \vdash_{Q} \phi; s \vdash_{\mathsf{E}} \Phi; \Sigma} \qquad \text{A-Elim} \ \frac{s \vdash_{\mathsf{A}} Q\,\phi, \Phi; \Sigma}{s \vdash_{Q} \phi; \Sigma \qquad s \vdash_{\mathsf{A}} \Phi; \Sigma}$$

Tableau Calculus

NICTA

Rules to expand U and R formulas

$$\text{U-Exp}\ \frac{s \vdash_Q (\phi \,\mathsf{U}\, \psi), \Phi; \Sigma}{s \vdash_Q \psi \lor (\phi \land \mathsf{X}\,(\phi \,\mathsf{U}\, \psi)), \Phi; \Sigma} \qquad \text{R-Exp}\ \frac{s \vdash_Q (\phi \,\mathsf{R}\, \psi), \Phi; \Sigma}{s \vdash_Q (\psi \land (\phi \lor \overline{\mathsf{X}}\,(\phi \,\mathsf{R}\, \psi))), \Phi; \Sigma}$$

Rules to simplify X formulas

$$\text{E-X-Simp}\ \frac{s \vdash_{\mathsf{E}} \mathsf{X}\,\phi_1, \ldots, \mathsf{X}\,\phi_n, \overline{\mathsf{X}}\,\psi_1, \ldots, \overline{\mathsf{X}}\,\psi_m; \Sigma}{s \vdash_{\mathsf{E}} Y\,(\phi_1 \land \cdots \land \phi_n \land \psi_1 \land \cdots \land \psi_m); \Sigma}$$

where $Y = \overline{\mathsf{X}}$ if n=0 else Y = X

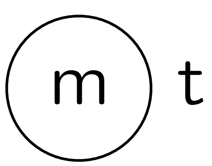
A-X-Simp: similary

Tableau Calculus

Rules to expand X-formulas

$$\text{E-}\overline{\mathsf{X}}\text{-Exp} \quad \frac{(m,t) \vdash_{\mathsf{E}} \overline{\mathsf{X}}\,\phi;\Sigma}{(n_1,u_1[t]) \vdash_{\mathsf{E}} \gamma_1[t] \wedge \phi;\Sigma \quad \cdots \quad (n_k,u_k[t]) \vdash_{\mathsf{E}} \gamma_k[t] \wedge \phi;\Sigma \quad (m,t) \vdash_{\mathsf{E}} \neg\gamma_1[t] \wedge \cdots \wedge \neg\gamma_k[t];\Sigma}$$

Intuitively



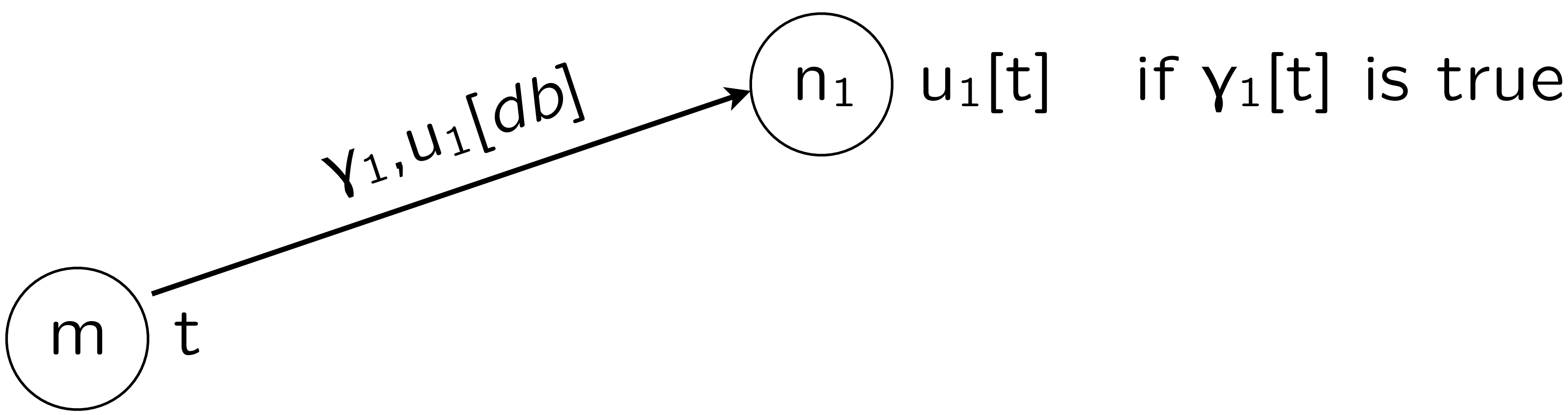
m,n	node
$\gamma[db]$	guard
$u[db]$	update-term

Tableau Calculus

Rules to expand X-formulas

$$\text{E-}\overline{\mathsf{X}}\text{-Exp}\quad \frac{(m,t) \vdash_{\mathsf{E}} \overline{\mathsf{X}}\,\phi;\Sigma}{(n_1,u_1[t]) \vdash_{\mathsf{E}} \gamma_1[t] \wedge \phi;\Sigma \quad \cdots \quad (n_k,u_k[t]) \vdash_{\mathsf{E}} \gamma_k[t] \wedge \phi;\Sigma \quad (m,t) \vdash_{\mathsf{E}} \neg\gamma_1[t] \wedge \cdots \wedge \neg\gamma_k[t];\Sigma}$$

Intuitively



m,n	node
γ[*db*]	guard
u[*db*]	update-term

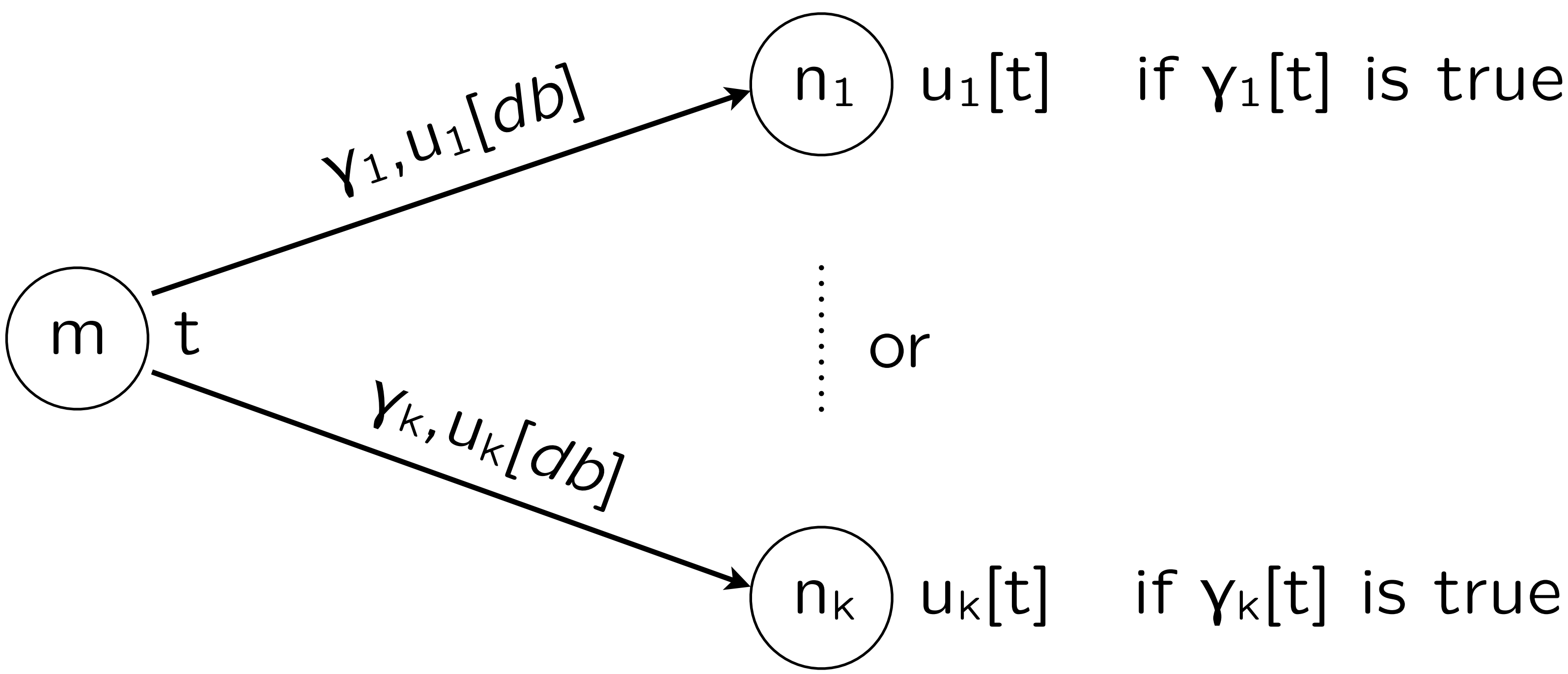
Tableau Calculus

NICTA

Rules to expand X-formulas

$$\text{E-}\overline{\mathsf{X}}\text{-Exp}\quad \frac{(m,t) \vdash_{\mathsf{E}} \overline{\mathsf{X}}\,\phi;\Sigma}{(n_1, u_1[t]) \vdash_{\mathsf{E}} \gamma_1[t] \wedge \phi;\Sigma \quad \cdots \quad (n_k, u_k[t]) \vdash_{\mathsf{E}} \gamma_k[t] \wedge \phi;\Sigma \quad (m,t) \vdash_{\mathsf{E}} \neg\gamma_1[t] \wedge \cdots \wedge \neg\gamma_k[t];\Sigma}$$

Intuitively



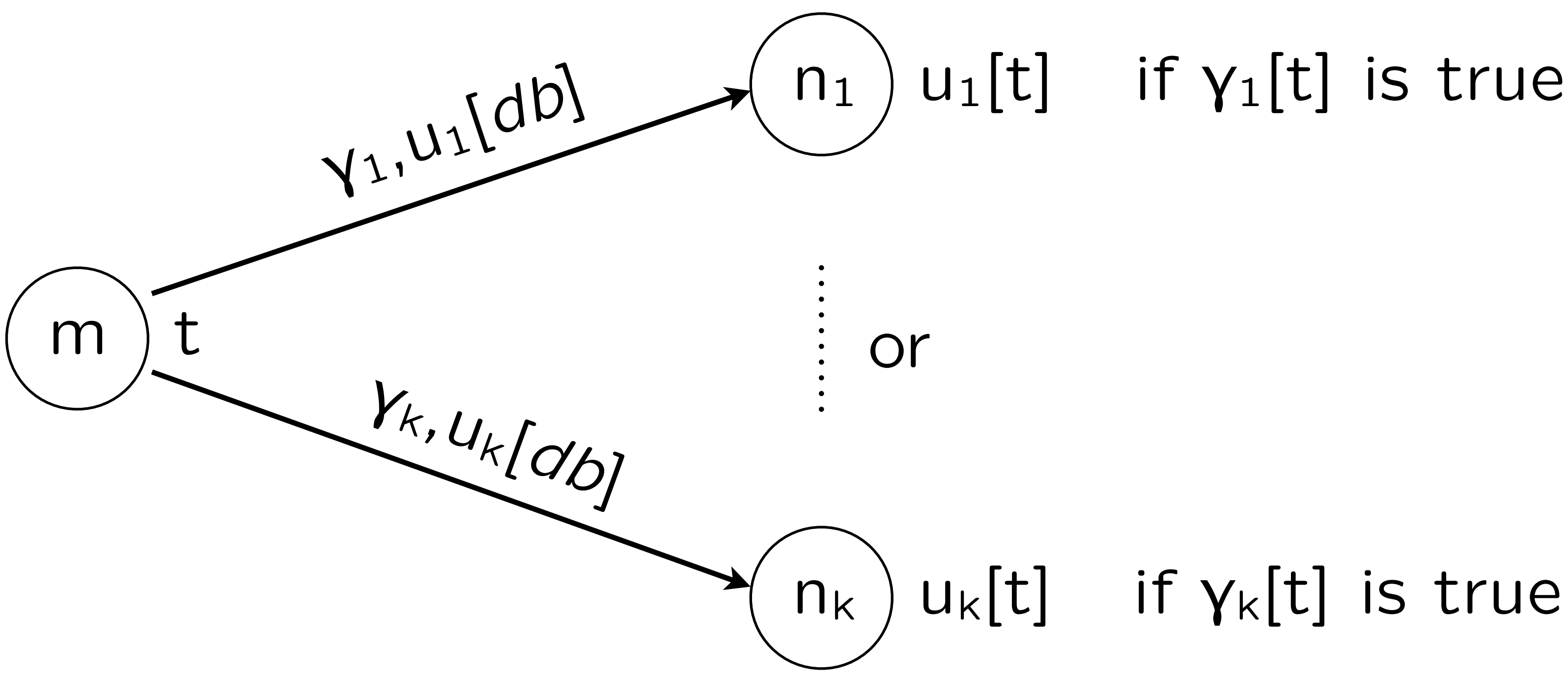
m,n	node
γ[*db*]	guard
u[*db*]	update-term

Tableau Calculus

Rules to expand X-formulas

E-$\overline{\mathsf{X}}$-Exp
$$\frac{(m,t) \vdash_{\mathsf{E}} \overline{\mathsf{X}}\,\phi;\Sigma}{(n_1,u_1[t]) \vdash_{\mathsf{E}} \gamma_1[t] \land \phi;\Sigma \quad \cdots \quad (n_k,u_k[t]) \vdash_{\mathsf{E}} \gamma_k[t] \land \phi;\Sigma \quad (m,t) \vdash_{\mathsf{E}} \neg\gamma_1[t] \land \cdots \land \neg\gamma_k[t];\Sigma}$$

Intuitively



or none of $\gamma_1[t]$, ..., $\gamma_k[t]$ is true

m,n	node
$\gamma[db]$	guard
$u[db]$	update-term

Tableau Calculus

NICTA

Rule for Closing branches

$$\text{Unsat}\ \frac{s_1 \vdash_{Q_1} \Phi_1; \cdots ; s_n \vdash_{Q_n} \Phi_n}{}$$

if all ϕ_i are **classical** formulas and $\phi_1 \wedge \cdots \wedge \phi_n$ is unsatisfiable

Tableau Calculus

Rule for Closing branches

$$\text{Unsat}\ \frac{s_1 \vdash_{Q_1} \Phi_1; \cdots; s_n \vdash_{Q_n} \Phi_n}{}$$

if all ϕ_i are **classical** formulas and $\phi_1 \wedge \cdots \wedge \phi_n$ is unsatisfiable

Theorem: soundness/completeness (decidability) for bounded model checking modulo FOL

Implementation and Experiments

NICTA

Fitzroy

Scala implementation of the above calculus + K-Induction

FOL-prover is currently Z3

"High-level" input language, type checker

Bounded model checking for paths up to given length n

E.g. **F** completed(*db*) and n=8 gives

Init → Pack → Stocktake → Pack → Invoice → Shipped → Paid

Init → Pack → Stocktake → Pack → Stocktake → Pack → Invoice → Shipped → Paid

Init → Pack → Stocktake → Pack → Invoice → Paid → Shipped

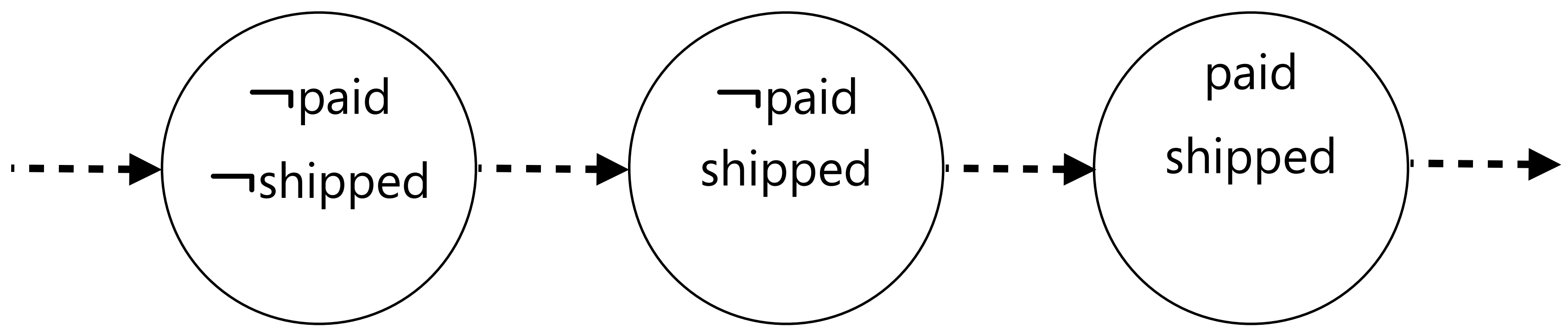
Init → Pack → Stocktake → Pack → Stocktake → Pack → Invoice → Paid → Shipped

(223 branches closed, 912 inferences, Z3 called 529 times, 30 sec)

Bounded Model Checking Example

(Recall queries are implicitly **E**-quantified)

$(\mathbf{F}\ \text{completed}(db)) \land (db.\text{shipped}{=}\text{true}\ \mathbf{R}\ db.\text{paid}{=}\text{false})$



The query is satisfiable because *db*.gold is possible

init → pack → stocktake → pack → invoice → shipped → paid

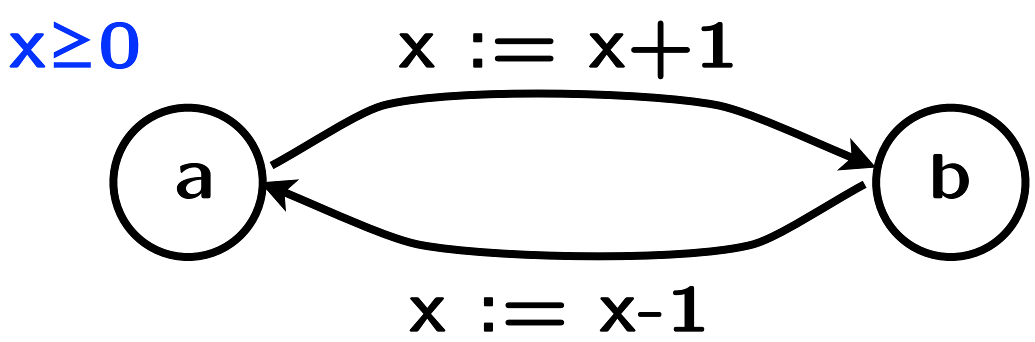
Proving Safety Properties with K-Induction

Question

Given a classical formula $\Phi[db]$,does $(I, s_0) \models$ **AG** $\Phi[db]$ hold,

for all interpretations I and all $s_0 \in$ Init?

K-induction [Sheeran et al 2000, deMoura et al 2003]



$\models$ **AG** $x \geq 0$

K = 0,1,2,... length of paths considered for inductive proofs

- 0-induction fails
- 1-induction goes through

  Base case: $x \geq 0 \wedge x' = x+1 \models x \geq 0 \wedge x' \geq 0$

  Step case, e.g.: $x \geq 0 \wedge x' = x-1 \wedge x' \geq 0 \wedge x'' = x'+1 \models x'' \geq 0$

Proving Safety Properties with K-Induction

NICTA

AG $(\forall i{:}\mathrm{Integer}.((0 \leq i \wedge i < \mathit{db}.\mathrm{nrStockItems}) \Rightarrow \mathit{db}.\mathrm{stock}[i].\mathrm{available} \geq 0))$

The number of available stock items is non-negative

Easy, after adding constraint on initial state

$\mathit{db}.\mathrm{nrStockItems} \geq 0 \wedge$

$(\forall i{:}\mathrm{Integer}.((0 \leq i \wedge i < \mathit{db}.\mathrm{nrStockItems}) \Rightarrow \mathit{db}.\mathrm{stock}[i].\mathrm{available} \geq 0))$

NB: *db*.nrStockItems is given *symbollically* - goes beyond propositional model checking

AG $((\mathit{db}.\mathrm{paid} = \mathrm{true} \wedge \mathit{db}.\mathrm{shipped} = \mathrm{false}) \Rightarrow \mathbf{F}\ \mathit{db}.\mathrm{shipped} = \mathrm{true})$

Paid but unshipped orders will be shipped eventually

Easy

Proving Safety Properties with K-Induction

NICTA

InRange predicate

inRange([1,4,0,5], 6)

$\forall$ l:List[Integer]. $\forall$n:Integer.

(inRange(l, n) $\Leftrightarrow$ (l = nil $\vee$ (0 $\leq$ head(l) $\wedge$ head(l) $<$ n $\wedge$ inRange(tail(l), n))))

AG inRange(*db*.open, *db*.nrStockItems)

All item numbers in the open list are in the range 0 ... *db*.nrStockItems-1

Provable with k=2 after adding constraint on inital state

db.nrStockItems $\geq$ 0 $\wedge$ inRange(db.open,db.nrStockItems)

Caveat

k=1 gives unprovable proof obligations where Z3 does not terminate.

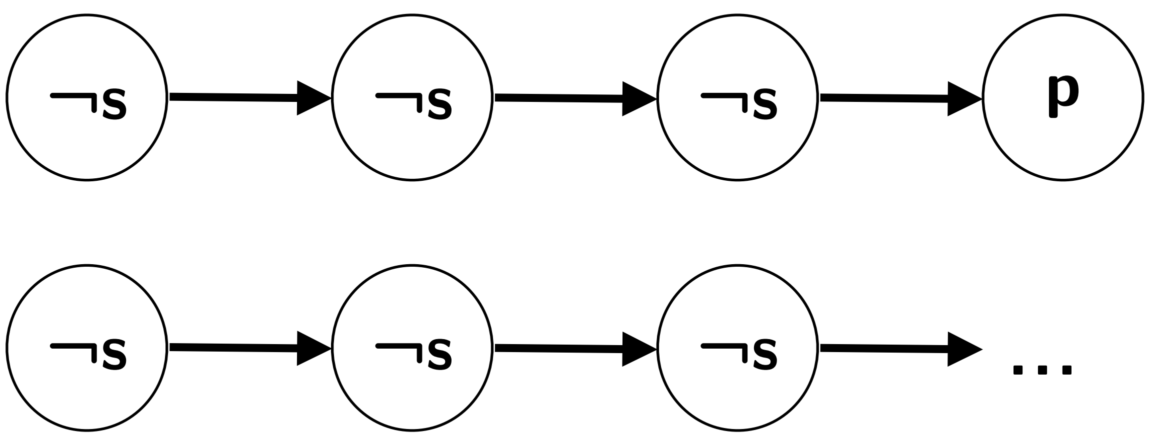
These proof obligations are *not quantifier-free*

Proving Safety Properties with K-Induction

$\mathbf{AG}\ ((db.\mathrm{gold} = \mathrm{false} \wedge db.\mathrm{shipped} = \mathrm{true}) \Rightarrow db.\mathrm{paid} = \mathrm{true})$

Follows from constraint

$db.\mathrm{gold} = \mathrm{false} \Rightarrow (db.\mathrm{shipped} = \mathrm{false}\ \mathbf{W}\ db.\mathrm{paid} = \mathrm{true}))$



But not provable because above constraint is ignored for K-Induction

Future Work

NICTA

Fighting the search space

Partial order reduction (gives many unprovable FOL-obligations)

Loop checks

Functional extensions

Nondeterministic assignments

db.nrRouters > 0
array[0..db.nrRouters] of Router
db.chosenRouter := i where $0 < i <$ db.nrRouters

Outputing refutations and models

Modules

First-order prover

Z3 incompleteness really hurts, e.g. can't show **LIST** $\not\models$ **4** $\in$ **[1,2,3]**

Integrate Beagle [B&Waldmann, CADE 2013]