

# “Global Discharge” – A Unified Rule for Loop Verification

Steffen Schlager

June 7, 2006

# Motivation & Introduction

- ▶ Global discharge rule based on fixed points
- ▶ Global discharge successfully used in EVT (Erlang Verification Tool)
- ▶ Unified rule for loop verification
- ▶ Understanding relation between inductive and invariant-based proofs
- ▶ Logic with fixed points is very expressive
- ▶ Fix point operators can be hidden for loop verification in KeY

## Signature

$\Sigma = (P, F, LVar, PVar, OrdVar, \alpha)$

## Formulas

Recursive definition of set of *Formulas* extended by:

- ▶  $\mu X.\varphi \in Formulas$ ,  $(\mu X.\varphi)^\kappa \in Formulas$   
 $\nu X.\varphi \in Formulas$ ,  $(\nu X.\varphi)^\kappa \in Formulas$   
for  $X \in PVar$ ,  $\kappa \in OrdVar$ ,  $\varphi \in Formulas$ , and  $X$  occurs only positively in  $\varphi$
- ▶  $\lambda x.\varphi \in Formulas$   
for  $x \in LVar$ ,  $\varphi \in Formulas$

# Dynamic Logic – Semantics

- ▶ For simplicity no free logical variables
- ▶ Semantics based on Kripke structures  $\mathcal{K} = (D, S, \rho)$
- ▶ Evaluation function  $val : Formula \rightarrow 2^S$
- ▶  $val(true) = S, val(false) = \emptyset$
- ▶  $val(P(t_1, \dots, t_n)) = \{s \in S \mid (I_s(t_1), \dots, I_s(t_n)) \in I_s(P)\}$
- ▶  $val(\neg\varphi) = S \setminus val(\varphi)$
- ▶  $val(\varphi \wedge \psi) = val(\varphi) \cap val(\psi)$

- ▶  $val(\mu X.\varphi) = lfp(\lambda X.\varphi)$
- ▶  $val(\nu X.\varphi) = gfp(\lambda X.\varphi)$

## Existence of lfp and gfp

Easy to show:

- ▶  $(2^S, \subseteq, \cup, \cap, \emptyset, S)$  forms a complete lattice
- ▶  $\lambda X.\varphi$  is monotonic (if  $X$  occurs only positively)

- ▶  $val(\mu X.\varphi) = lfp(\lambda X.\varphi)$
- ▶  $val(\nu X.\varphi) = gfp(\lambda X.\varphi)$

## Existence of lfp and gfp

Easy to show:

- ▶  $(2^S, \subseteq, \cup, \cap, \emptyset, S)$  forms a complete lattice
- ▶  $\lambda X.\varphi$  is monotonic (if  $X$  occurs only positively)

Knaster-Tarski Theorem:

If  $\lambda X.\varphi$  is monotonic  $(2^S, \subseteq, \cup, \cap, \emptyset, S)$  and is a complete lattice then  $lfp(\lambda X.\varphi)$  and  $gfp(\lambda X.\varphi)$  exist.

# Approximation of Fixed Points

- ▶ If  $f$  is monotonic on a lattice then the least fixed point can be obtained by iterating  $f$  on the bottom element  $\perp$  (here  $\emptyset$ ).

# Approximation of Fixed Points

- ▶ If  $f$  is monotonic on a lattice then the least fixed point can be obtained by iterating  $f$  on the bottom element  $\perp$  (here  $\emptyset$ ).
- ▶ Length of iteration is in general transfinite but *bound* by the cardinality  $\kappa$  of the lattice.

$$lfp(f) = \bigcup_{\alpha < \kappa} f^\alpha(\emptyset)$$

# Approximation of Fixed Points

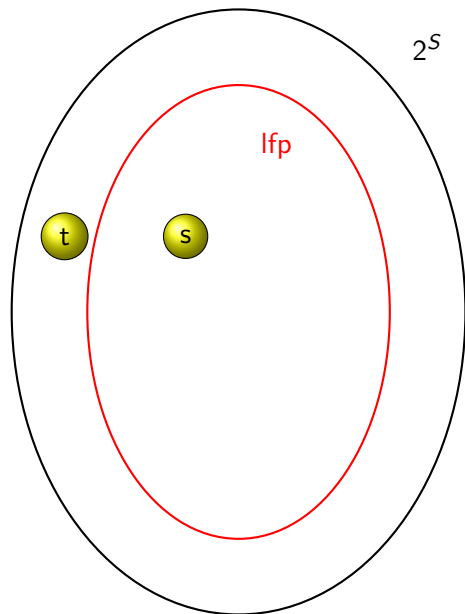
- ▶ If  $f$  is monotonic on a lattice then the least fixed point can be obtained by iterating  $f$  on the bottom element  $\perp$  (here  $\emptyset$ ).
- ▶ Length of iteration is in general transfinite but *bound* by the cardinality  $\kappa$  of the lattice.

$$lfp(f) = \bigcup_{\alpha < \kappa} f^\alpha(\emptyset)$$

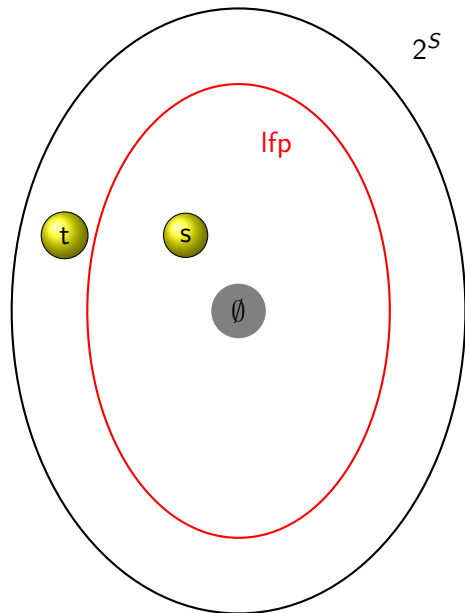
- ▶ Similar for the greatest fixed point

$$gfp(f) = \bigcap_{\alpha < \kappa} f^\alpha(S)$$

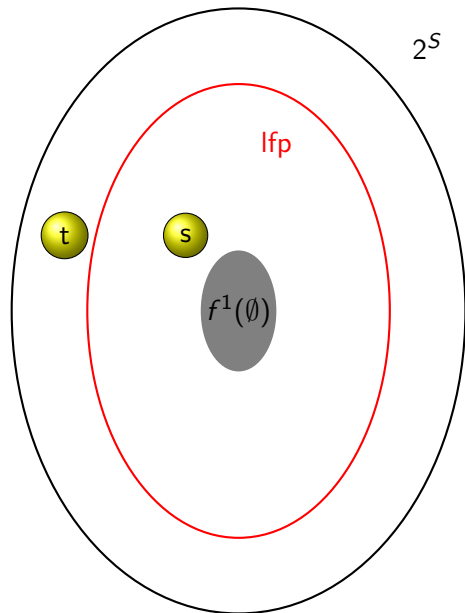
# Approximation of Least Fixed Points



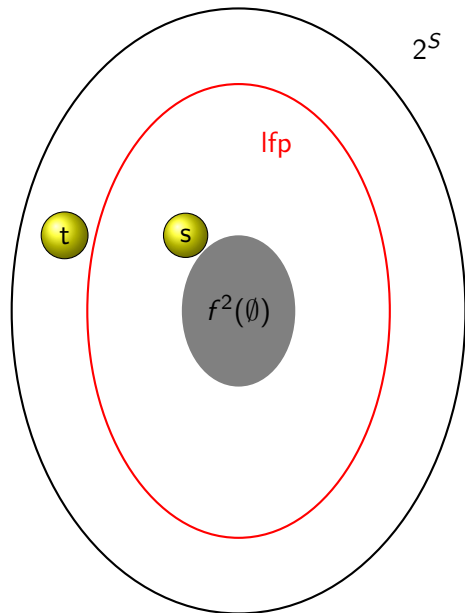
# Approximation of Least Fixed Points



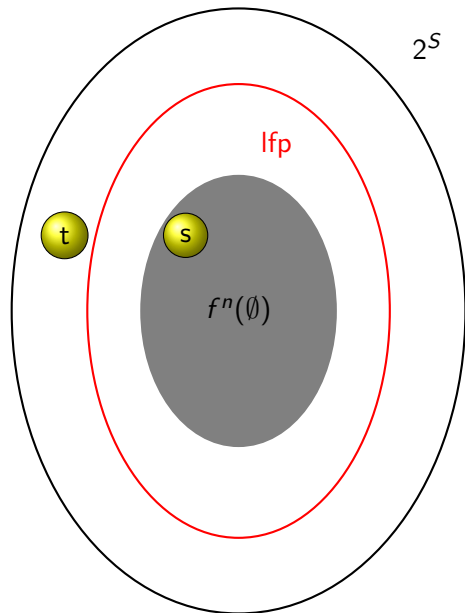
# Approximation of Least Fixed Points



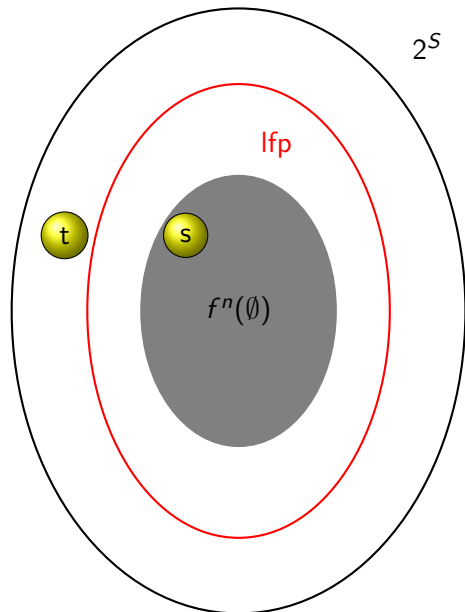
# Approximation of Least Fixed Points



# Approximation of Least Fixed Points



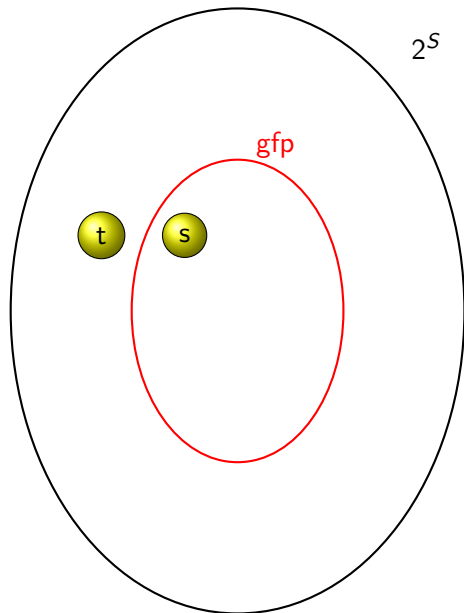
# Approximation of Least Fixed Points



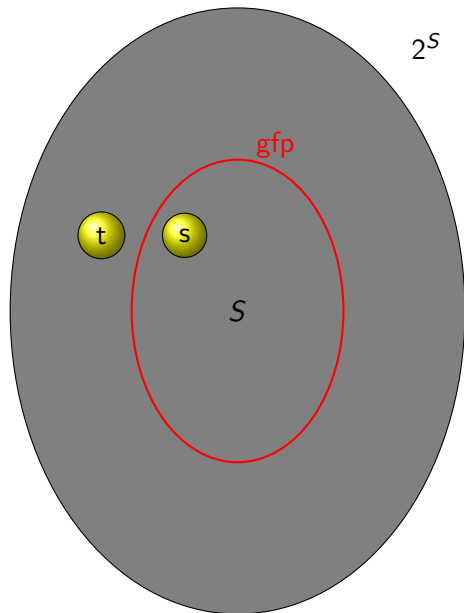
Intuitive meaning of least fixed points:

“finite looping”

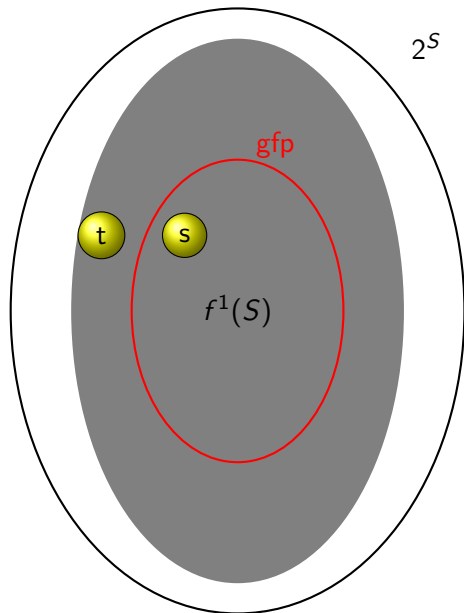
# Approximation of Greatest Points



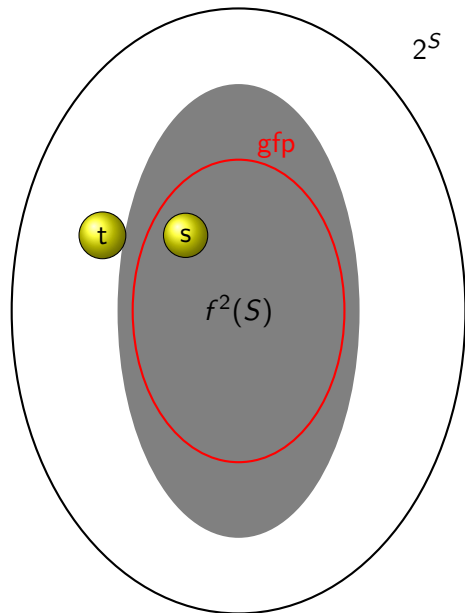
# Approximation of Greatest Points



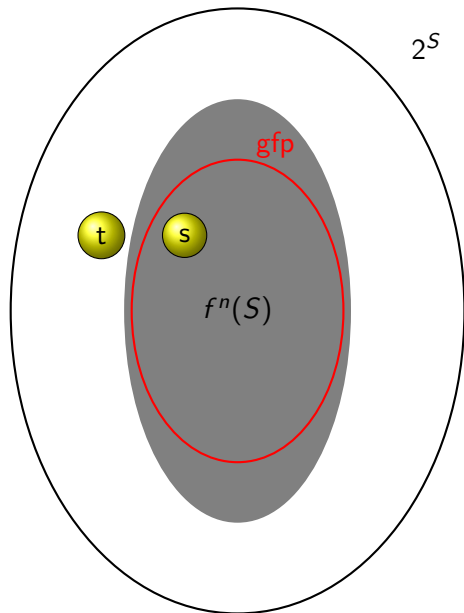
# Approximation of Greatest Points



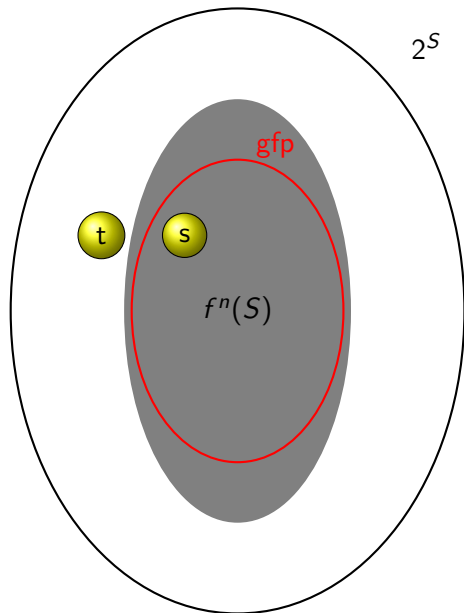
# Approximation of Greatest Points



# Approximation of Greatest Points



# Approximation of Greatest Points



Intuitive meaning of greatest fixed points:  
"infinite looping"

# Least or Greatest Fixed Points?

## Data types

- ▶ finite data types (e.g. natural numbers, lists, trees):  $\mu$
- ▶ infinite data types (e.g. streams, infinite lists):  $\nu$

# Least or Greatest Fixed Points?

## Data types

- ▶ finite data types (e.g. natural numbers, lists, trees):  $\mu$
- ▶ infinite data types (e.g. streams, infinite lists):  $\nu$

## Example

- ▶  $\mu X. \lambda n. (n \doteq 0 \vee \exists n'. n' \doteq n - 1 \wedge X(n'))$

# Least or Greatest Fixed Points?

## Data types

- ▶ finite data types (e.g. natural numbers, lists, trees):  $\mu$
- ▶ infinite data types (e.g. streams, infinite lists):  $\nu$

## Example

- ▶  $\mu X. \lambda n. (n \doteq 0 \vee \exists n'. n' \doteq n - 1 \wedge X(n'))$

## Temporal properties

- ▶ Liveness properties:  $\mu$
- ▶ Safety properties:  $\nu$

# Least or Greatest Fixed Points?

## Data types

- ▶ finite data types (e.g. natural numbers, lists, trees):  $\mu$
- ▶ infinite data types (e.g. streams, infinite lists):  $\nu$

## Example

- ▶  $\mu X. \lambda n. (n \doteq 0 \vee \exists n'. n' \doteq n - 1 \wedge X(n'))$

## Temporal properties

- ▶ Liveness properties:  $\mu$
- ▶ Safety properties:  $\nu$

## Example

- ▶  $\mu X. \varphi \vee \langle \alpha \rangle X$

# Least or Greatest Fixed Points?

## Data types

- ▶ finite data types (e.g. natural numbers, lists, trees):  $\mu$
- ▶ infinite data types (e.g. streams, infinite lists):  $\nu$

## Example

- ▶  $\mu X. \lambda n. (n \doteq 0 \vee \exists n'. n' \doteq n - 1 \wedge X(n'))$

## Temporal properties

- ▶ Liveness properties:  $\mu$
- ▶ Safety properties:  $\nu$

## Example

- ▶  $\mu X. \varphi \vee \langle \alpha \rangle X$
- ▶  $\nu X. \varphi \wedge [\alpha] X$

# Sequent Rules for $\mu$

$$\frac{(\mu X.\varphi)^k \vdash \Delta}{\mu X.\varphi \vdash \Delta} \text{ApproxL}$$

# Sequent Rules for $\mu$

$$\frac{(\mu X.\varphi)^\kappa \vdash \Delta}{\mu X.\varphi \vdash \Delta} \text{ApproxL}$$

Soundness:

We assume  $(\mu X.\varphi)^\kappa \vdash \Delta$  is valid, i.e. for any valuation  $\beta$  of  $\kappa$   
 $val(\beta, (\mu X.\varphi)^\kappa) \subseteq val(\Delta)$ .

# Sequent Rules for $\mu$

$$\frac{(\mu X.\varphi)^\kappa \vdash \Delta}{\mu X.\varphi \vdash \Delta} \text{ApproxL}$$

Soundness:

We assume  $(\mu X.\varphi)^\kappa \vdash \Delta$  is valid, i.e. for any valuation  $\beta$  of  $\kappa$   
 $val(\beta, (\mu X.\varphi)^\kappa) \subseteq val(\Delta)$ .

We further assume  $\mu X.\varphi \vdash \Delta$  is not valid, i.e. for some  $\beta$  there  
exists  $s$  with  $s \in val(\beta, \mu X.\varphi)$  and  $s \notin val(\Delta)$ .

# Sequent Rules for $\mu$

$$\frac{(\mu X.\varphi)^\kappa \vdash \Delta}{\mu X.\varphi \vdash \Delta} \text{ApproxL}$$

Soundness:

We assume  $(\mu X.\varphi)^\kappa \vdash \Delta$  is valid, i.e. for any valuation  $\beta$  of  $\kappa$   
 $val(\beta, (\mu X.\varphi)^\kappa) \subseteq val(\Delta)$ .

We further assume  $\mu X.\varphi \vdash \Delta$  is not valid, i.e. for some  $\beta$  there  
exists  $s$  with  $s \in val(\beta, \mu X.\varphi)$  and  $s \notin val(\Delta)$ .

From the semantics of  $\mu$  follows that there is some  $\beta'$  such that  
 $s \in val(\beta', (\mu X.\varphi)^\kappa)$  which is a contradiction to the assumption.



# Sequent Rules for $\mu$

$$\frac{\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta}{(\mu X.\varphi)^{\kappa} \vdash \Delta} \text{UnfoldL}$$

# Sequent Rules for $\mu$

$$\frac{\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta}{(\mu X.\varphi)^{\kappa} \vdash \Delta} \text{UnfoldL}$$

Soundness:

We assume  $\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta$  is valid, i.e. for any valuation  $\beta$  of  $\kappa', \kappa$   $\text{val}(\beta, \varphi\{(\mu X.\varphi)^{\kappa'} / X\}) \subseteq \text{val}(\Delta)$ .

We further assume  $(\mu X.\varphi)^{\kappa} \vdash \Delta$  is not valid, i.e. for some  $\beta$  there exists  $s$  with  $s \in \text{val}(\beta, (\mu X.\varphi)^{\kappa})$  and  $s \notin \text{val}(\Delta)$ .

# Sequent Rules for $\mu$

$$\frac{\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta}{(\mu X.\varphi)^\kappa \vdash \Delta} \text{UnfoldL}$$

Soundness:

We assume  $\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta$  is valid, i.e. for any valuation  $\beta$  of  $\kappa', \kappa$   $val(\beta, \varphi\{(\mu X.\varphi)^{\kappa'} / X\}) \subseteq val(\Delta)$ .

We further assume  $(\mu X.\varphi)^\kappa \vdash \Delta$  is not valid, i.e. for some  $\beta$  there exists  $s$  with  $s \in val(\beta, (\mu X.\varphi)^\kappa)$  and  $s \notin val(\Delta)$ .

**Case 1,  $\beta(\kappa) = 0$ :**  $val(\beta, (\mu X.\varphi)^\kappa) = \emptyset$ , contradiction!

# Sequent Rules for $\mu$

$$\frac{\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta}{(\mu X.\varphi)^{\kappa} \vdash \Delta} \text{UnfoldL}$$

Soundness:

We assume  $\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta$  is valid, i.e. for any valuation  $\beta$  of  $\kappa', \kappa$   $\text{val}(\beta, \varphi\{(\mu X.\varphi)^{\kappa'} / X\}) \subseteq \text{val}(\Delta)$ .

We further assume  $(\mu X.\varphi)^{\kappa} \vdash \Delta$  is not valid, i.e. for some  $\beta$  there exists  $s$  with  $s \in \text{val}(\beta, (\mu X.\varphi)^{\kappa})$  and  $s \notin \text{val}(\Delta)$ .

**Case 1,  $\beta(\kappa) = 0$ :**  $\text{val}(\beta, (\mu X.\varphi)^{\kappa}) = \emptyset$ , contradiction!

**Case 2,  $\beta(\kappa) = v + 1$ :** we set  $\beta(\kappa') = v$  and obtain a contradiction.

# Sequent Rules for $\mu$

$$\frac{\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta}{(\mu X.\varphi)^{\kappa} \vdash \Delta} \text{UnfoldL}$$

Soundness:

We assume  $\kappa' < \kappa, \varphi\{(\mu X.\varphi)^{\kappa'} / X\} \vdash \Delta$  is valid, i.e. for any valuation  $\beta$  of  $\kappa', \kappa$   $val(\beta, \varphi\{(\mu X.\varphi)^{\kappa'} / X\}) \subseteq val(\Delta)$ .

We further assume  $(\mu X.\varphi)^{\kappa} \vdash \Delta$  is not valid, i.e. for some  $\beta$  there exists  $s$  with  $s \in val(\beta, (\mu X.\varphi)^{\kappa})$  and  $s \notin val(\Delta)$ .

**Case 1,  $\beta(\kappa) = 0$ :**  $val(\beta, (\mu X.\varphi)^{\kappa}) = \emptyset$ , contradiction!

**Case 2,  $\beta(\kappa) = v + 1$ :** we set  $\beta(\kappa') = v$  and obtain a contradiction.

**Case 3,  $\beta(\kappa)$  is a limit ordinal (e.g.  $\omega$ ):**

From the semantics of approximated fix points for limit ordinals (which is the union of all approximated fix points with  $\kappa < \omega$ ) follows that there is a non-limit ordinal  $\kappa''$  such that  $s \in val(\beta, (\mu X.\varphi)^{\kappa''})$  and then the previous case applies.

# Sequent Rules for $\mu$

$$\frac{\vdash \varphi\{(\mu X.\varphi)/X\}, \Delta}{\vdash \mu X.\varphi, \Delta} \text{UnfoldR}$$

# Sequent Rules for $\nu$

$$\frac{\vdash (\nu X.\varphi)^{\kappa}, \Delta}{\vdash \nu X.\varphi, \Delta} \text{ApproxR}$$

# Sequent Rules for $\nu$

$$\frac{\vdash (\nu X.\varphi)^\kappa, \Delta}{\vdash \nu X.\varphi, \Delta} \textit{ApproxR}$$

$$\frac{\kappa' < \kappa \vdash \varphi\{(\nu X.\varphi)^{\kappa'} / X\}, \Delta}{\vdash (\nu X.\varphi)^\kappa, \Delta} \textit{UnfoldR}$$

# Sequent Rules for $\nu$

$$\frac{\vdash (\nu X.\varphi)^\kappa, \Delta}{\vdash \nu X.\varphi, \Delta} \text{ApproxR}$$

$$\frac{\kappa' < \kappa \vdash \varphi\{(\nu X.\varphi)^{\kappa'} / X\}, \Delta}{\vdash (\nu X.\varphi)^\kappa, \Delta} \text{UnfoldR}$$

$$\frac{\varphi\{(\nu X.\varphi) / X\} \vdash \Delta}{\nu X.\varphi \vdash \Delta} \text{UnfoldL}$$





# Global Discharge Rule – Basic Idea

$$\begin{array}{c}
 \vdots \\
 \hline
 \kappa'' < \kappa', \kappa' < \kappa \vdash \sigma'(\sigma(\Delta_1)), \sigma'(\sigma((\nu X.\varphi)^\kappa)) \\
 \hline
 \vdots \\
 \hline
 \kappa' < \kappa \vdash \sigma(\Delta_1), \sigma((\nu X.\varphi)^\kappa), \Delta_n \\
 \hline
 \vdots \\
 \hline
 \vdash \Delta_4, (\nu X.\varphi)^\kappa \qquad \vdash \Delta_5, (\nu X.\varphi)^\kappa \quad \text{Unfold} \\
 \hline
 \vdash \Delta_2 \qquad \vdash \Delta_3, (\nu X.\varphi)^\kappa \\
 \hline
 \vdash \Delta_1, (\nu X.\varphi)^\kappa \quad \text{Approx} \\
 \hline
 \vdash \Delta_1, \nu X.\varphi
 \end{array}$$

# Global Discharge Rule – Substitution Instance

## Definition (Substitution Instance)

A node in a proof labelled with  $\Gamma_D \vdash \Delta_D$  is a *substitution instance* of a predecessor node labelled with  $\Gamma_C \vdash \Delta_C$  if there is a substitution  $\sigma$  such that

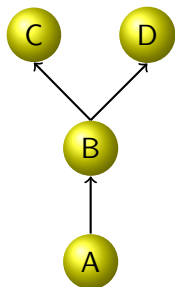
- ▶  $\sigma(\Gamma_C) \subseteq \Gamma_D$
- ▶  $\sigma(\Delta_C) \subseteq \Delta_D$

## Definition (Loop)

A *loop* is a sequence  $(N_0, \dots, N_k)$  of nodes in the proof tree such that  $N_k$  is a leaf node which is a substitution instance of  $N_0$ .

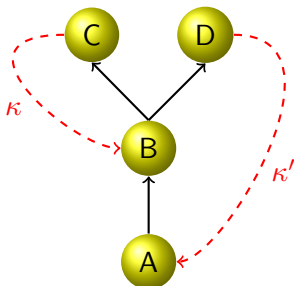


# Global Discharge Rule – Progress Condition



- ▶ Identify set  $SC = \{BC, ABD, ABCD\}$  of strongly connected sub-graphs.
- ▶ In every  $s \in SC$  there is a loop such that an ordinal  $\kappa$  *decreases* and all other ordinals are *preserved*, i.e. are not increased.

# Global Discharge Rule – Progress Condition



- ▶ Identify set  $SC = \{BC, ABD, ABCD\}$  of strongly connected sub-graphs.
- ▶ In every  $s \in SC$  there is a loop such that an ordinal  $\kappa$  *decreases* and all other ordinals are *preserved*, i.e. are not increased.

# Loop Verification Using Global Discharge in KeY

Idea – Understanding loops as fixed points:

- ▶ if  $\alpha$  terminates normally

$$\langle \text{while } (\epsilon) \{ \alpha \} \rangle \varphi \leftrightarrow \mu X. (\neg \epsilon \rightarrow \varphi) \wedge (\epsilon \rightarrow \langle \alpha \rangle X)$$

$$[\text{while } (\epsilon) \{ \alpha \}] \varphi \leftrightarrow \nu X. (\neg \epsilon \rightarrow \varphi) \wedge (\epsilon \rightarrow [\alpha] X)$$

# Loop Verification Using Global Discharge in KeY

Idea – Understanding loops as fixed points:

- ▶ if  $\alpha$  terminates normally

$$\langle \text{while } (\epsilon) \{ \alpha \} \rangle \varphi \leftrightarrow \mu X. (\neg \epsilon \rightarrow \varphi) \wedge (\epsilon \rightarrow \langle \alpha \rangle X)$$

$$[\text{while } (\epsilon) \{ \alpha \}] \varphi \leftrightarrow \nu X. (\neg \epsilon \rightarrow \varphi) \wedge (\epsilon \rightarrow [\alpha] X)$$

- ▶ if  $\alpha$  may terminate abruptly:

$$\langle \pi \text{ while } (\epsilon) \{ \alpha \} \omega \rangle \varphi \leftrightarrow$$

$$\mu X. ((\neg \epsilon \rightarrow \langle \pi \omega \rangle \varphi) \vee (\epsilon \rightarrow \langle \pi \alpha \omega \rangle_{AT} \varphi)) \wedge (\epsilon \rightarrow \langle \alpha \rangle_{NT} X)$$

$$[\pi \text{ while } (\epsilon) \{ \alpha \} \omega] \varphi \leftrightarrow$$

$$\mu X. ((\neg \epsilon \rightarrow [\pi \omega] \varphi) \vee (\epsilon \rightarrow [\pi \alpha \omega]_{AT} \varphi)) \wedge (\epsilon \rightarrow [\alpha]_{NT} X)$$



# Justification

## Justification

Let  $\rho_\pi : S \rightarrow S$  be the partial evaluation function for a program  $\pi$ .  
For a loop  $\pi = \text{while } (\epsilon) \{ \alpha \}$  we obtain

$$\rho_\pi(s) = \begin{cases} s & \text{if } s \notin \text{val}(\epsilon) \\ \perp & \text{if } s \in \text{val}(\epsilon) \text{ and } \perp = \rho_\alpha(s) \\ \rho_\pi(s') & \text{if } s \in \text{val}(\epsilon) \text{ and } s' = \rho_\alpha(s) \end{cases}$$

which is a recursively defined monotone function and its least fixed point is the semantics of the loop.

If  $\pi$  does not terminate for any initial state the least fixed point is  $\lambda s. \perp$ .



# Loop Verification Using Global Discharge in KeY

## Idea

- ▶ Unfolding of loops corresponds to unfolding of fixed points

# Loop Verification Using Global Discharge in KeY

## Idea

- ▶ Unfolding of loops corresponds to unfolding of fixed points
- ▶ Use ordinal variables to record unfolding

# Loop Verification Using Global Discharge in KeY

## Idea

- ▶ Unfolding of loops corresponds to unfolding of fixed points
- ▶ Use ordinal variables to record unfolding
- ▶ Use statement of derivability (including updates) instead of substitution instance

$$\mathcal{U}(\sigma(\Gamma_C \rightarrow \Delta_C)) \vdash \Gamma_D \rightarrow \Delta_D$$

Demo

# Summary & Future Work

## Summary

- ▶ One rule for inductive and invariant (co-inductive) proofs
- ▶ DL-invariant rule and induction rule are derivable
- ▶ Works well for recursive operations

# Summary & Future Work

## Summary

- ▶ One rule for inductive and invariant (co-inductive) proofs
- ▶ DL-invariant rule and induction rule are derivable
- ▶ Works well for recursive operations

## Future Work

- ▶ Implementation (minor thesis by Daniel Bruns)
- ▶ Engineering right abstraction from failed attempts of global discharge
- ▶ Verification of temporal properties

$$\vdash \alpha : \nu X. \varphi \wedge \langle - \rangle X$$

there is a path such that  $\varphi$  always holds

