Program Analysis and Verification

0368-4479

Noam Rinetzky

Lecture 5: Rely/Guarantee Reasoning

Slides credit: Roman Manevich, Mooly Sagiv, Eran Yahav

While + Concurrency

Abstract syntax:

$$a ::= n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2$$

$$b ::= \mathtt{true} \mid \mathtt{false}$$
$$\mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \land b_2$$

$$S ::= x := a \mid \mathtt{skip} \mid S_1; S_2$$
$$\mid \mathtt{if}\ b\ \mathtt{then}\ S_1\ \mathtt{else}\ S_2$$
$$\mid \mathtt{while}\ b\ \mathtt{do}\ S$$
$$\mid S_1 \| \ldots \| S_n$$

While + concurrency: structural semantics

$$\frac{\langle S_i, s\rangle \rightarrow \langle S_i', s' \rangle\rangle}{\langle\ \langle S_1 \| \dots\ \ \| S_n, s\rangle \rightarrow \rangle\ \langle\ S_1 \| \dots\ \| S'_i \| \dots \| S_n, s'\rangle \Rightarrow} \qquad i=1..n$$

$$\frac{\langle S_i, s\rangle \rightarrow \rangle\ s' \ \rangle}{\langle\ \langle S_1 \| \dots\ \ \| S_n, s\rangle \rightarrow \rangle\ \langle\ S_1 \| \dots\ \| \mathbf{done} \| \dots \| S_n, s'\rangle} \qquad i=1..n$$

Proofs

$$\dfrac{\dfrac{\cdots}{\{ P_1 \}\ S_1\ \{ Q_2 \}} \qquad \dfrac{\cdots}{\{ P_2 \}\ S_2\ \{ Q_2 \}}}{\{ P_1 \wedge P_2 \}\ S_1 \parallel S_2\ \{ Q_1 \wedge Q_2 \}} \quad \cdots$$

Challenge:
Interference

Axiomatic Semantics (Hoare Logic)

- Disjoint parallelism
- Global invariant
- Owicky – Gries [PhD. '76]
- Rely/Guarantee [Jones.]

$$\frac{\ldots}{\{\,P\,\}\ S_1 \parallel S_2\ \{\,Q\,\}} \quad \ldots$$

Rely / Guarantee

- Aka assume Guarantee
- Cliff Jones
- Main idea: Modular capture of interference
  - Compositional proofs

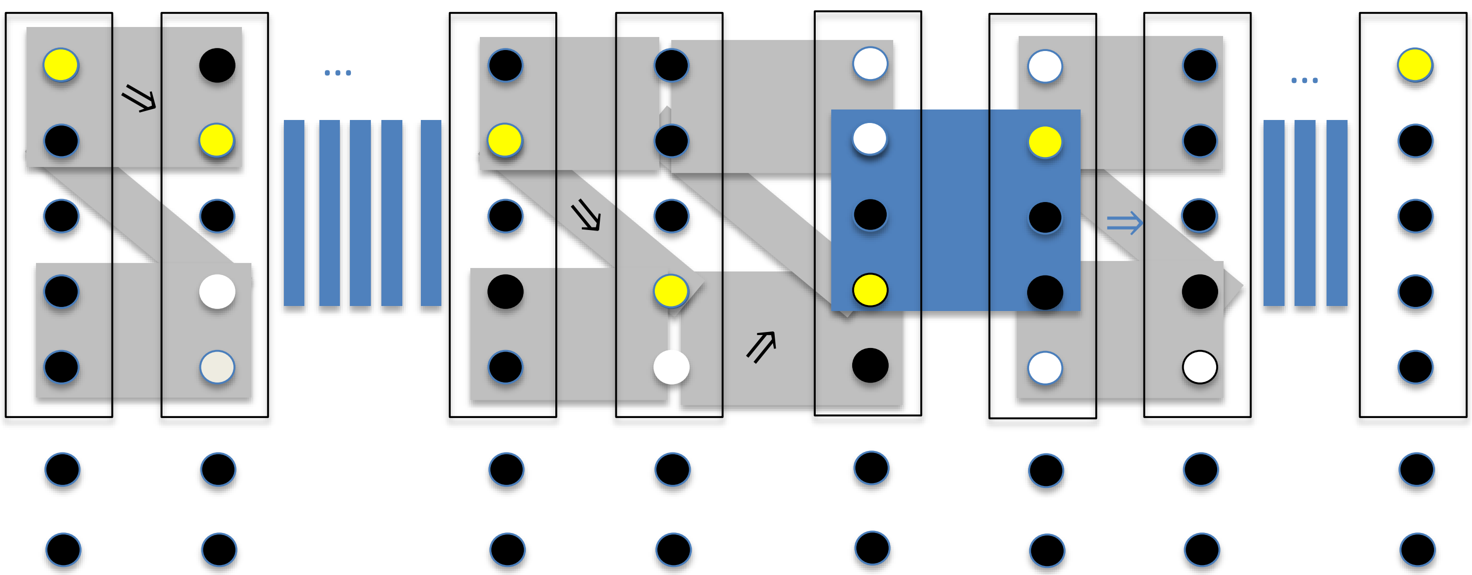
Commands as relations

- It is convenient to view the meaning of commands as relations between pre-states and post-states
- In {P} C {Q}
  - P is a one state predicate
  - Q is a two-state predicate
    - Recall auxiliary variables
- Example
  - {true} x := x + 1 {x= $\underline{x}$ + 1}

Intuition: Rely Guarantee

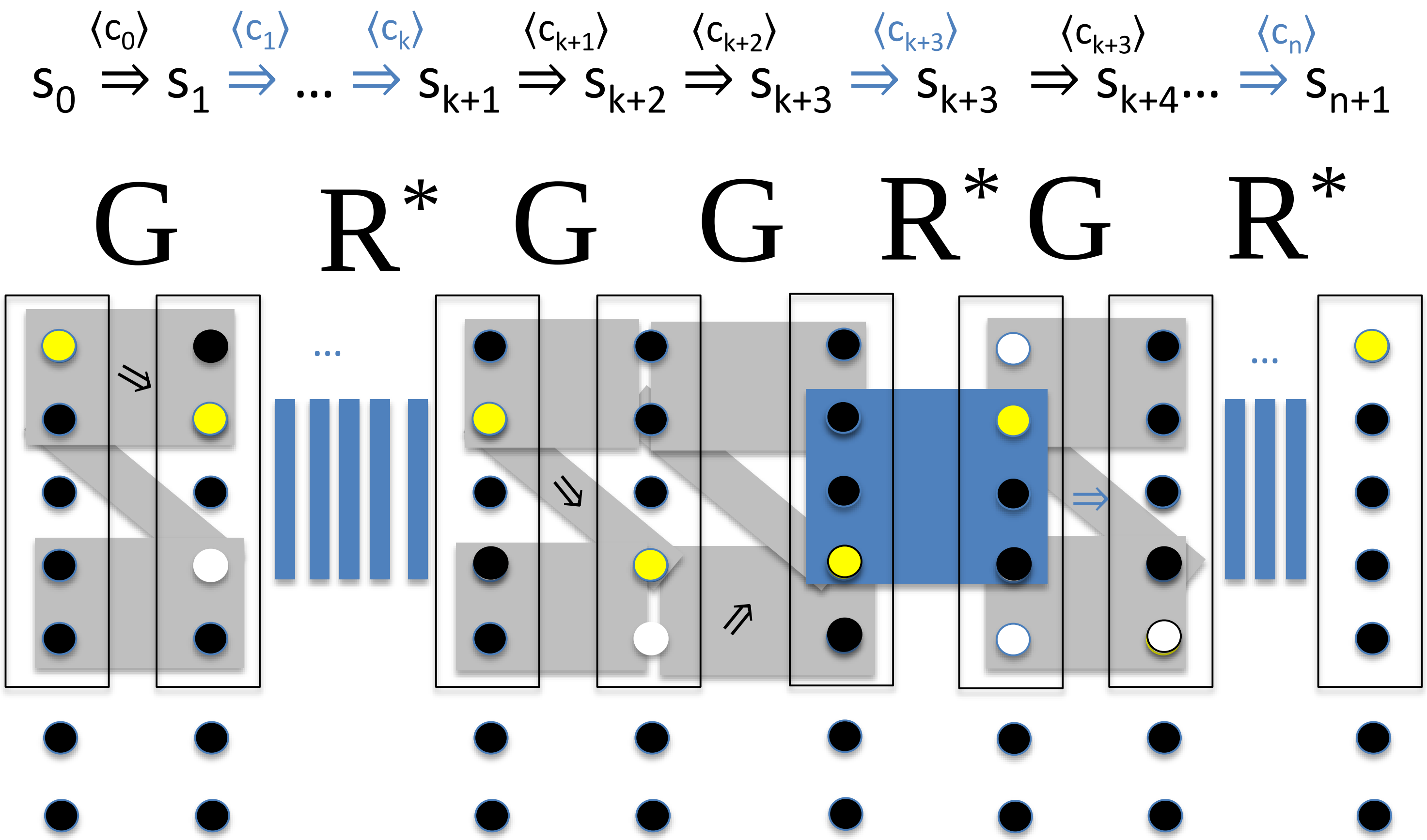
- Thread-view

$$s_0 \xRightarrow{\langle c_0\rangle} s_1 \xRightarrow{\langle c_1\rangle} \ldots \xRightarrow{\langle c_k\rangle} s_{k+1} \xRightarrow{\langle c_{k+1}\rangle} s_{k+2} \xRightarrow{\langle c_{k+2}\rangle} s_{k+3} \xRightarrow{\langle c_{k+3}\rangle} s_{k+3} \xRightarrow{\langle c_{k+3}\rangle} s_{k+4} \ldots \xRightarrow{\langle c_n\rangle} s_{n+1}$$



Intuition: Rely Guarantee

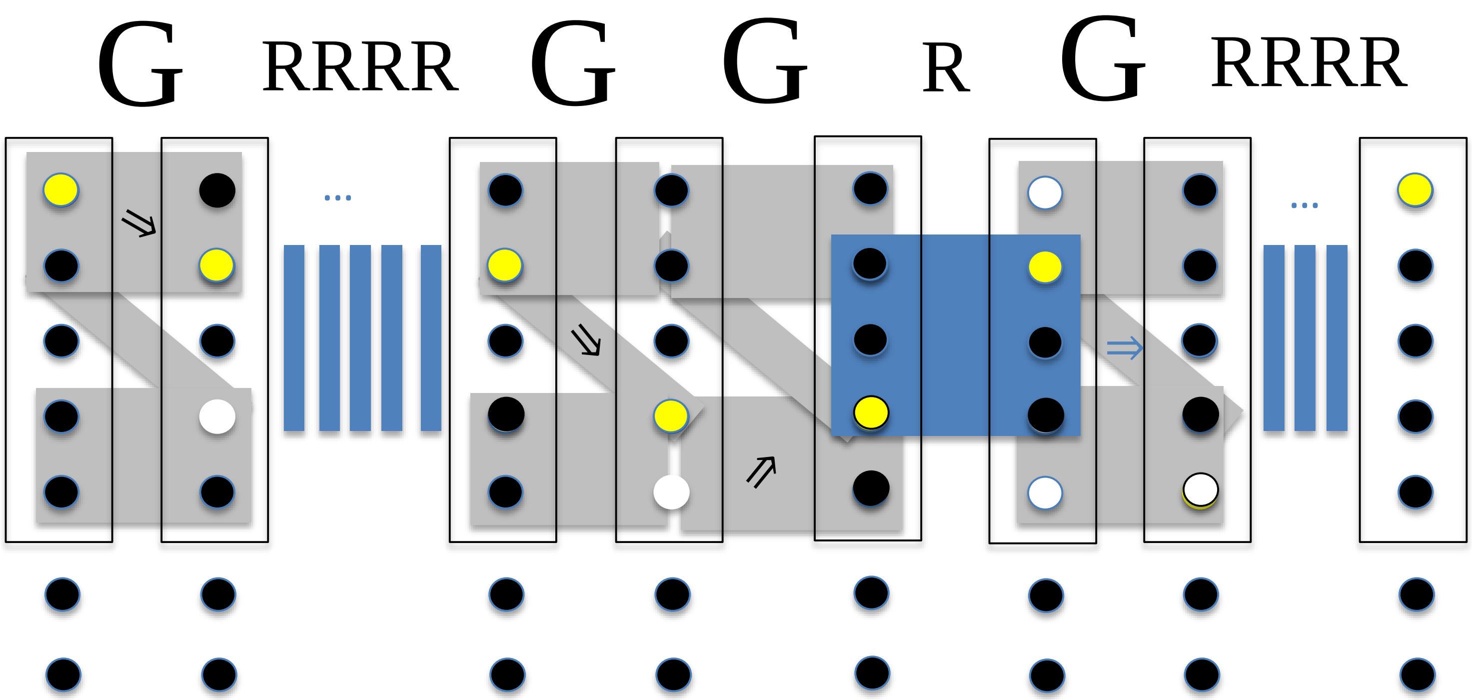
- Thread-view



Intuition: Rely Guarantee

- Thread-view

$$s_0 \overset{\langle c_0 \rangle}{\Rightarrow} s_1 \overset{\langle c_1 \rangle}{\Rightarrow} \ldots \overset{\langle c_k \rangle}{\Rightarrow} s_{k+1} \overset{\langle c_{k+1} \rangle}{\Rightarrow} s_{k+2} \overset{\langle c_{k+2} \rangle}{\Rightarrow} s_{k+3} \overset{\langle c_{k+3} \rangle}{\Rightarrow} s_{k+3} \overset{\langle c_{k+3} \rangle}{\Rightarrow} s_{k+4} \ldots \overset{\langle c_n \rangle}{\Rightarrow} s_{n+1}$$



Intuition (again)

Hoare: $\{\,P\,\}\ S\ \{\,Q\,\} \sim \{\mathrm{P}\}\overbrace{\Rightarrow\Rightarrow\Rightarrow\Rightarrow}^{\mathrm{C}}\{\mathrm{Q}\}$

R/G: $\mathrm{R,G} \vdash \{\,P\,\}\ S\ \{\,Q\,\} \sim \{\mathrm{P}\}\overbrace{\xRightarrow[\mathrm{R}]{}\xRightarrow[\mathrm{G}]{}\xRightarrow[\mathrm{R}]{}\xRightarrow[\mathrm{R}]{}\xRightarrow[\mathrm{G}]{}\xRightarrow[\mathrm{G}]{}\xRightarrow[\mathrm{G}]{}}^{\mathrm{C}}\{\mathrm{Q}\}$

Relational Post-Conditions

- meaning of commands a relations between pre-states and post-states

- {P} C {Q}
  - P is a one state predicate
  - Q is a two-state predicate

- Example
  - {true} x := x + 1 {x= $\underline{x}$ + 1}

Goal: Parallel Composition

$$\frac{\begin{array}{c} R \vee G_2, G_1 \vdash \{\, P \,\}\, S_1 \,\{\, Q \,\} \\ R \vee G_1, G_2 \vdash \{\, P \,\}\, S_2 \,\{\, Q \,\} \end{array}}{R, G_1 \vee G_2 \vdash \{\, P \,\}\, S_1 \| S_2 \,\{\, Q \,\}} \quad \text{(PAR)}$$

Relational Post-Conditions

- meaning of commands a relations between pre-states and post-states

- Option I: {P} C {Q}
  - P is a one state predicate
  - Q is a two-state predicate

- Example
  - {true} x := x + 1 {x= $\underline{x}$ + 1}

From one- to two-state relations

- $p(\underline{\sigma}, \sigma) = p(\sigma)$
- $\underline{p}(\underline{\sigma}, \sigma) = p(\underline{\sigma})$
- A single state predicate p is **preserved** by a two-state relation R if
  - $\underline{p} \wedge R \Rightarrow p$
  - $\forall \underline{\sigma}, \sigma\text{: } p(\underline{\sigma}) \wedge R(\underline{\boldsymbol{\sigma}}, \sigma) \Rightarrow p(\sigma)$

Operations on Relations

- $(P;Q)(\underline{\sigma}, \sigma)=\exists\tau:P(\underline{\sigma}, \tau) \wedge Q(\tau, \sigma)$
- $ID(\underline{\sigma}, \sigma)= (\underline{\sigma}=\sigma)$
- $R^{*}=ID\vee R \vee(R;R) \vee(R;R;R) \vee\ldots \vee$

Formulas

- $ID(x) = (\underline{x} = x)$
- $ID(p) = (\underline{p} \Leftrightarrow p)$
- $\text{Preserve}(p) = \underline{p} \Rightarrow p$

Informal Semantics

- $c \models (p, R, G, Q)$
  - For every state $\underline{\sigma}$ such that $\underline{\sigma} \models p$:
    - Every execution of c on state $\underline{\sigma}$ with (potential) interventions which satisfy R results in a state σ such that $(\underline{\sigma}, \sigma) \models Q$
    - The execution of every atomic sub-command of c on any possible intermediate state satisfies G

Informal Semantics

- $c \models (p, R, G, Q)$
  - For every state $\underline{\sigma}$ such that $\underline{\sigma} \models p$:
    - Every execution of c on state $\underline{\sigma}$ with (potential) interventions which satisfy R results in a state σ such that $(\underline{\sigma}, \sigma) \models Q$
    - The execution of every atomic sub-command of c on any possible intermediate state satisfies G
- $c \models [p, R, G, Q]$
  - For every state $\underline{\sigma}$ such that $\underline{\sigma} \models p$:
    - Every execution of c on state $\underline{\sigma}$ with (potential) interventions which satisfy R must terminate in a state σ such that $(\underline{\sigma}, \sigma) \models Q$
    - The execution of every atomic sub-command of c on any possible intermediate state satisfies G

A Formal Semantics

- Let $[\![C]\!]^R$ denotes the set of quadruples $<\sigma_1, \sigma_2, \sigma_3, \sigma_4>$ s.t. that when c executes on σ_1 with potential interferences by R it yields an intermediate state σ_2 followed by an intermediate state σ_3 and a final state σ_4
  - as usual $\sigma_4=\perp$ when c does not terminate

- $[\![C]\!]^R = \{<\sigma_1, \sigma_2, \sigma_3, \sigma_4> : \exists \sigma: <\sigma1, \sigma> \models R \wedge$
  $(<C, \sigma> \Rightarrow^* \sigma_2 \wedge \sigma_2 = \sigma_3 = \sigma_4 \vee$
  $\exists \sigma', C' : <C, \sigma> \Rightarrow^* <C', \sigma'>$
  $\wedge ((\sigma_2 = \sigma_1 \vee \sigma_2 = \sigma) \wedge (\sigma_3 = \sigma \vee \sigma_3 = \sigma') \wedge \sigma_4 = \perp)$
  $\vee <\sigma', \sigma_2, \sigma_3, \sigma_4> \in [\![C']\!]^R)$

- $c \models (p, R, G, Q)$
  - For every $<\sigma_1, \sigma_2, \sigma_3, \sigma_4> \in [\![C]\!]^R$ such that $\sigma_1 \models p$
    - $<\sigma_2, \sigma_3> \models G$
    - If $\sigma4 \neq \perp$: $<\sigma1, \sigma4> \models Q$

Simple Examples

- $X := X + 1 \models (\text{true}, X=\underline{X}, X=\underline{X}+1 \vee X=\underline{X}, X=\underline{X}+1)$
- $X := X + 1 \models (X \geq 0, X \geq \underline{X}, X>0 \vee X=\underline{X}, X>0)$
- $X := X + 1 ; Y := Y + 1 \models (X \geq 0 \wedge Y \geq 0, X \geq \underline{X} \wedge Y \geq \underline{Y}, G, X>0 \wedge Y>0)$

Inference Rules

- Define $c \vdash (p, R, G, Q)$ by structural induction on c
- Soundness
  - If $c \vdash (p, R, G, Q)$ then $c \models (p, R, G, Q)$

Atomic Command

$$\frac{\{p\}\ c\ \{Q\}}{\text{atomic}\ \{c\} \vdash (p,\ \text{preserve}(p),\ Q \vee ID,\ Q)}\quad (\text{Atomic})$$

Conditional Critical Section

$$\frac{\{p \wedge b\}\ c\ \{Q\}}{\text{await } b \text{ then } c \vdash (p,\ \text{preserve}(p),\ Q \vee ID,\ Q)} \quad \text{(Critical)}$$

Sequential Composition

$$c_1 \vdash (p_1, R, G, Q_1)$$

$$c_2 \vdash (p_2, R, G, Q_2)$$

$$Q_1 \Rightarrow p_2$$

$$\frac{}{c_1 \,;\, c_2 \vdash (p_1, R, G, (Q_1; R^*; Q_2))} \quad \text{(SEQ)}$$

Conditionals

$$\frac{c_1 \vdash (p \wedge b_1, R, G, Q) \quad p \wedge b \wedge R^* \Rightarrow b_1 \qquad c_2 \vdash (p \wedge b_2, R, G, Q) \quad p \wedge \neg b \wedge R^* \Rightarrow b_2}{\text{if atomic } \{b\} \text{ then } c_1 \text{ else } c_2 \vdash (p, R, G, Q)} \quad \text{(IF)}$$

Loops

$$\frac{c \vdash (j \wedge b_1, R, G, j) \quad j \wedge b \wedge R^* \Rightarrow b_1 \quad R \Rightarrow \mathrm{Preserve}(j)}{\text{while atomic } \{b\} \text{ do } c \vdash (j, R, G, \neg b \wedge j)} \quad \text{(WHILE)}$$

Refinement

$$\frac{c \vdash (p, R, G, Q) \qquad p' \Rightarrow p \qquad Q \Rightarrow Q' \qquad R' \Rightarrow R \qquad G \Rightarrow G'}{c \vdash (p', R', G', Q')} \text{(REFINE)}$$

Parallel Composition

$$\frac{\begin{array}{c} c_1 \vdash (p_1, R_1, G_1, Q_1) \\ c_2 \vdash (p_2, R_2, G_2, Q_2) \\ G_1 \Rightarrow R_2 \\ G_2 \Rightarrow R_1 \end{array}}{c_1 \,||\, c_2 \vdash (p_1 \wedge p_1, (R_1 \wedge R2), (G_1 \vee G_2), Q)} \quad \text{(PAR)}$$

where $Q = (Q_1 ; (R_1 \wedge R_2)^*; Q_2) \vee (Q_2 ; (R_1 \wedge R_2)^*; Q_1)$

Issues in R/G

- Total correctness is trickier
- Restrict the structure of the proofs
  - Sometimes global proofs are preferable
- Many design choices
  - Transitivity and Reflexivity of Rely/Guarantee
  - No standard set of rules
- Suitable for designs

Example: the FINDP algorithm

Problem:

given an array $v[1..n]$ and a predicate P,
find the smallest r such that $P(v[r])$ holds.

A sequential specification in Hoare logic:

$\{ \forall i . P(v[i]) \text{ is defined } \}$

findp

$\{ (r = n + 1 \land \forall i . \neg P(v[i])) \lor (1 \le r \le n \land P(v[r]) \land \forall i < r. \neg P(v[i])) \}$

An R/G specification of the *findp* algorithm:

$$findp \models (\, pre\, ,\ v = \underline{v} \land r = \underline{r}\, ,\ \text{True}\, ,\ post\,)$$

Guarantee: this thread can modify the state arbitrarily.

Rely: other threads cannot modify v or r.

where *pre* and *post* are as above.

Example: a concurrent FINDP algorithm

Idea:

- partition the array,
- multiple processes search concurrently, one process per partition.

Simple way: `even` and `odd` processes.

Naive concurrency: each process searches a partition, calculates the final result as the minimum of the result of the `even` and `odd` processes.

Problem: can perform worse than sequential (why?)

Example: a concurrent FINDP algorithm

Idea:

- partition the array,
- multiple processes search concurrently, one process per partition.

Simple way: `even` and `odd` processes.

Naive concurrency: each process searches a partition, calculates the final result as the minimum of the result of the `even` and `odd` processes.

Problem: can perform worse than sequential (why?)

Communicating processes:

- introduce a (shared) variable `top` that records the lowest index that satisifies P found so far;
- each thread checks at each iteration that it did not go past `top`.

Example: specification of concurrent FINDP

FindpWorker $\models$

pre: $\forall\ \mathtt{i} \in$ partition . P($\mathtt{v[i]}$)) is defined

rely: $\mathtt{v} = \underline{\mathtt{v}} \land \mathtt{top} \leq \underline{\mathtt{top}}$

guar: $\mathtt{top} = \underline{\mathtt{top}} \lor \mathtt{top} < \underline{\mathtt{top}} \land$ P($\mathtt{v[top]}$)

post: $\forall\ \mathtt{i} \in$ partition, $\mathtt{i} \leq \mathtt{top} \Rightarrow \neg$P($\mathtt{v[i]}$)

Rely: other threads cannot modify `v` and can only decrement `top`.

Guarantee: the other threads are guaranteed that, if this thread updates `top`, the new value is smaller than the older and is such that P(`v[top]`)holds.

It is then possible to prove that two FindpWorkers, running in parallel, satisfy the specificationof Findp described two slides ago.

(modulo setting up the partitions appropriately and copying the final value from top to r)