CS256/Spring 2008 — Lecture #16

Zohar Manna

References for further reading:

- Volume III of Manna & Pnueli, Chapter 1

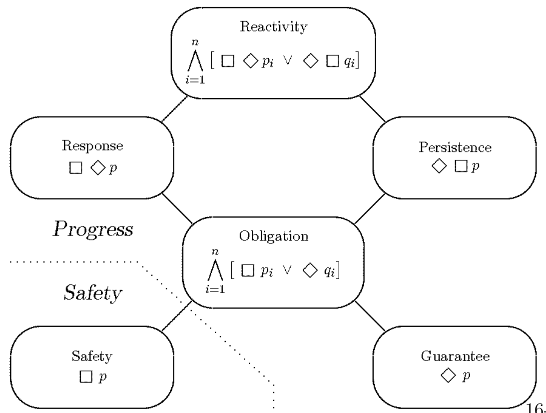
- Zohar Manna and Amir Pnueli. "Completing the Temporal Picture." In *Theoretical Computer Science Journal*, 83(1), 1991, pp. 97–130.

References are available from Zohar Manna's web page, `http://theory.stanford.edu/~zm/`; look at the class web site for a link to the initial chapters of Volume III.

Volume III
Progress

Progress properties:
Temporal logic plays a more prominent role and fairness becomes important.

Property hierarchy:



Response under Justice
(Chapter 1)

Progress Properties

We will consider deductive methods to prove response properties (which are also applicable to obligation and guarantee properties since these are subclasses)

Response properties are those properties that can be expressed by a formula of the formula of the form

$$\Box\Diamond p$$

for a past formula p.

Response formulas

The verification rules presented assume that the response property is expressed by a response formula

$$p \Rightarrow \Diamond q$$

for past formulas p and q.

Note:

- Response formula expresses a response property because of the equivalence

$$p \Rightarrow \Diamond q \quad \sim \quad \Box\Diamond((\neg p)\mathcal{B}q)$$

- Every response property can be expressed by a response formula due to the equivalence

$$\Box\Diamond q \quad \sim \quad \mathrm{T} \Rightarrow \Diamond q$$

Overview

We consider the simple case where p, q are assertions.

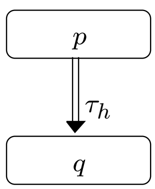
The proof of a response property

$$p \Rightarrow \Diamond q$$

often relies on the identification of one or more so-called helpful transitions. We consider three cases:

1. Rule RESP-J
   (single-step response under justice)

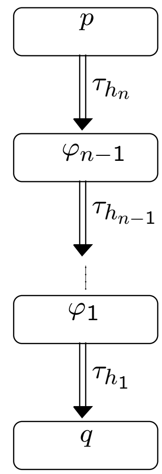
   A single helpful transition τ_h suffices to establish the property



Overview (Cont'd)

2. Rule CHAIN-J
   (chain rule under justice)

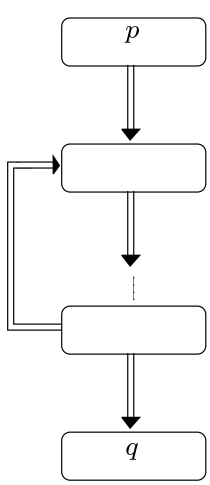
   A fixed number of helpful transitions (independent of the value of variables) suffices to establish the property



Overview (Cont'd)

3. Rule WELL-J
   (well-founded response under justice)

   The number of helpful transitions required to establish the property is unbounded



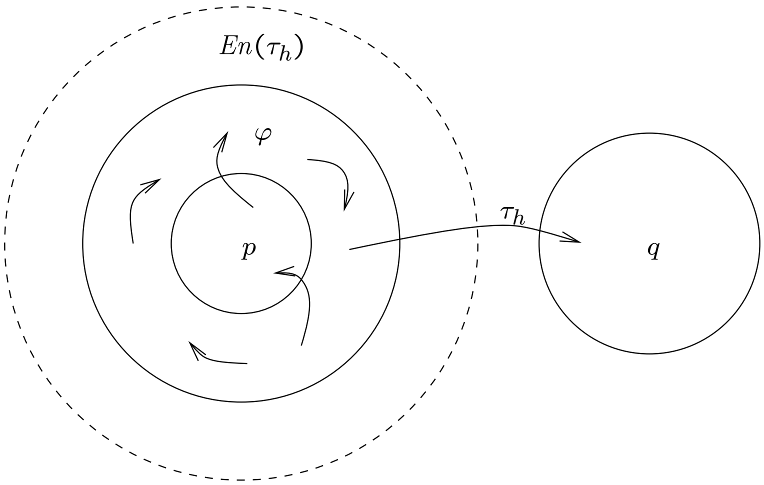
Overview (Cont'd)

In all cases we will be able to use verification diagrams to represent the proof.

In practice, verification diagrams are often the preferred way to prove progress properties, because they represent the temporal structure of the program relative to the property.

Single-step rule (Motivation)

$p \Rightarrow \Diamond q$



Justice requirement: it is not the case that a just transition is continuously enabled but never taken.

Single-step rule

For assertions p, q, φ, and helpful transition $\tau_h \in \mathcal{J}$,

$$\begin{array}{ll} \text{J1.} & p \rightarrow q \vee \varphi \\ \text{J2.} & \{\varphi\}\ \mathcal{T}\ \{q \vee \varphi\} \\ \text{J3.} & \{\varphi\}\ \tau_h\ \{q\} \\ \text{J4.} & \varphi \rightarrow En(\tau_h) \\ \hline & p \Rightarrow \Diamond q \end{array}$$

Premise J2 requires all transitions to preserve φ (or establish q, in which case we are done).

Premise J4 ensures that the helpful transition τ_h will be continuously enabled.

It ensures, by the justice requirement, that τ_h will eventually be taken.

Premise J3 guarantees that it will establish q.

Single-step rule (Cont'd)

In practice, this rule is not very useful:

Very few properties rely on just a single helpful transition.

This leads to the CHAIN rule, where we have several intermediate properties.

Useful rules

- Monotonicity:
$$\frac{p \Rightarrow q \quad q \Rightarrow \Diamond r \quad r \Rightarrow t}{p \Rightarrow \Diamond t}$$

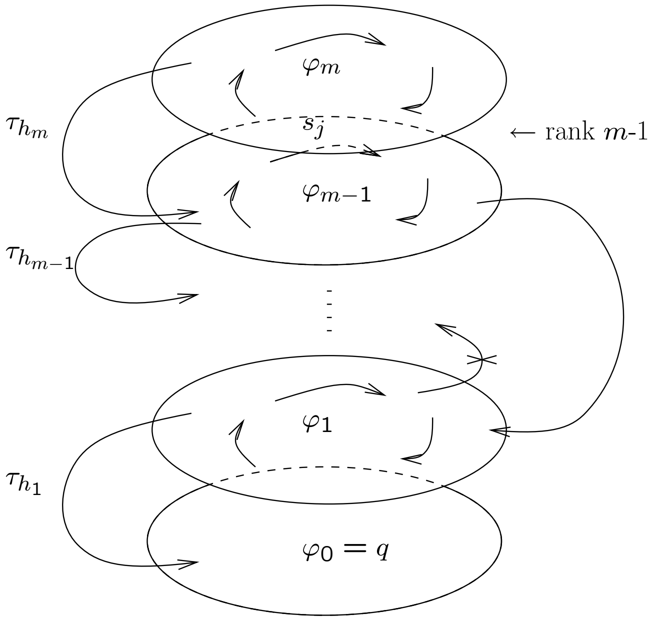
- Reflexivity:
$$p \Rightarrow \Diamond p$$

- Transitivity:
$$\frac{p \Rightarrow \Diamond q \quad q \Rightarrow \Diamond r}{p \Rightarrow \Diamond r}$$

- Case analysis:
$$\frac{p \Rightarrow \Diamond r \quad q \Rightarrow \Diamond r}{(p \vee q) \Rightarrow \Diamond r}$$

Chain rule (Motivation)

$p \Rightarrow \Diamond\, q$



For state s_j: let φ_i be the intermediate formula with the smallest i such that $s_j \Vdash \varphi_i$. Then i is the *rank* of s_j.

Chain rule

For assertions $p,\ q = \varphi_0$ and $\varphi_1, \ldots,\ \varphi_m$ and helpful transitions $\tau_{h_1}, \ldots,\ \tau_{h_m} \in \mathcal{J}$

J1. $p \rightarrow \bigvee_{j=0}^{m} \varphi_j$

$$\left.\begin{array}{ll} \text{J2.} & \{\varphi_i\}\mathcal{T}\left\{\bigvee_{j\le i} \varphi_j\right\} \\ \text{J3.} & \{\varphi_i\}\tau_{h_i}\left\{\bigvee_{j< i} \varphi_j\right\} \\ \text{J4.} & \varphi_i \rightarrow En(\tau_{h_i}) \end{array}\right\} \text{for } i = 1, \ldots, m$$

$p \Rightarrow \Diamond\, q$

J2: rank never increases
J3: rank decreases

Chain rule (Cont'd)

It is our task to find the intermediate assertions $\varphi_m, \ldots, \varphi_1$.

Premise J2 ensures that all transitions either preserve the current assertion or move down to a lower-ranked assertion.

Premise J4 ensures that the helpful transition τ_{h_i} is enabled for φ_i, which makes it impossible to stay in φ_i forever, by the justice requirement.

Premise J3 guarantees that the helpful transition moves down to a strictly lower-ranked assertion.

Since premises J2–J4 hold for every $1 \le i \le m$, this ensures that $\varphi_0 = q$ will be reached eventually.

Verification Diagrams

Nodes: labeled by assertions φ_i

Terminal node φ_0

Edges: labeled by transitions

single-lined
(represents a regular transition)

double-lined
(represents a helpful transition)

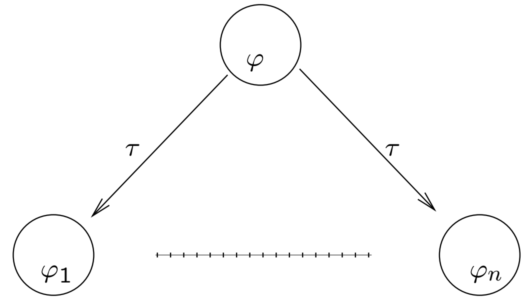
Chain diagram

well-formedness conditions:

- weakly acyclic in $\longrightarrow$:
  if $\varphi_i \longrightarrow \varphi_j$ then $i \geq j$
- acyclic in $\Longrightarrow$:
  if $\varphi_i \Longrightarrow \varphi_j$ then $i > j$
- every nonterminal node has a double edge departing from it
- no transition can label both a single and a double edge departing from the same node.

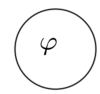
Chain diagram: verification conditions

1. single τ-edges



$$\{\varphi\}\tau\{\varphi \vee \varphi_1 \vee \ldots \vee \varphi_n\}$$

nonterminal node with no outgoing τ-edges:

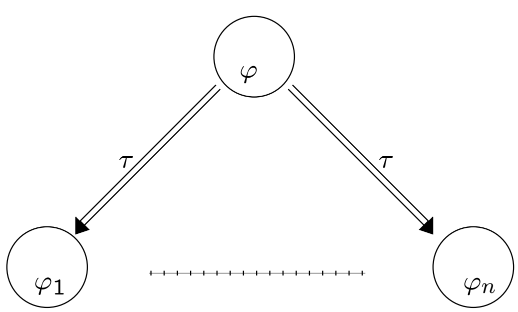


$\{\varphi\}\tau\{\varphi\}$

Note: No Verification Condition for terminal node.

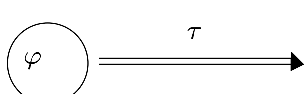
Chain diagram verification conditions (Cont'd)

2. double τ-edges



$$\{\varphi\}\tau\{\varphi_1 \vee \ldots \vee \varphi_n\}$$

3. enabling condition



$\varphi \rightarrow En(\tau)$

Chain diagrams: validity

A chain diagram is P-valid
if all the verification conditions associated with the diagram are P-valid.

Claim: A P-valid chain diagram establishes that

$$\bigvee_{j=0}^{m} \varphi_j \Rightarrow \Diamond\, \varphi_0$$

is P-valid.

With $\quad p \rightarrow \bigvee_{j=0}^{m} \varphi_j \quad$ and $\quad \varphi_0 \rightarrow q$,
we can conclude the P-validity of

$$\boxed{p \Rightarrow \Diamond\, q}$$

Example: Program mux-pet1 (Fig. 3.4)

(Peterson's Algorithm for mutual exclusion)

local y_1, y_2: **boolean** **where** $y_1 = \text{F}, y_2 = \text{F}$
s : **integer** **where** $s = 1$

$$P_1 :: \left[\begin{array}{l} \ell_0: \ \textbf{loop forever do} \\ \left[\begin{array}{ll} \ell_1: & \textbf{noncritical} \\ \ell_2: & (y_1,\ s) := (\text{T},\ 1) \\ \ell_3: & \textbf{await } (\neg y_2) \vee (s \neq 1) \\ \ell_4: & \textbf{critical} \\ \ell_5: & y_1 := \text{F} \end{array} \right] \end{array} \right]$$

$||$

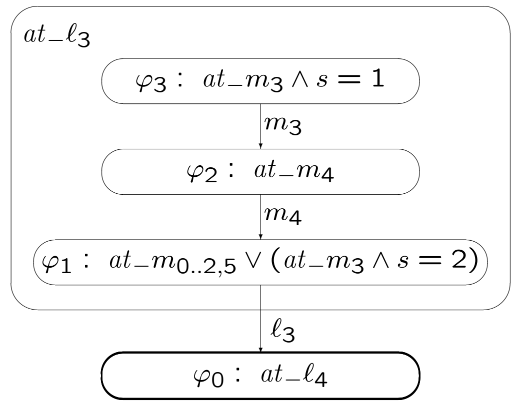
$$P_2 :: \left[\begin{array}{l} m_0: \ \textbf{loop forever do} \\ \left[\begin{array}{ll} m_1: & \textbf{noncritical} \\ m_2: & (y_2,\ s) := (\text{T},\ 2) \\ m_3: & \textbf{await } (\neg y_1) \vee (s \neq 2) \\ m_4: & \textbf{critical} \\ m_5: & y_2 := \text{F} \end{array} \right] \end{array} \right]$$

Example: Accessibility for MUX-PET1

In Chapter 3 of the SAFETY book we established 1-bounded overtaking, expressed by

$$at\_\ell_3 \ \Rightarrow \ \neg at\_m_4 \ \mathcal{W} \ at\_m_4 \ \mathcal{W} \ \neg at\_m_4 \ \mathcal{W} \ at\_\ell_4$$

for MUX-PET1 with the following WAIT-diagram



Example (Cont'd)

We now want to establish accessibility, expressed by

$$\boxed{at\_\ell_3 \Rightarrow \Diamond \ at\_\ell_4}$$

Since the two properties seem similar we would like to transform the WAIT diagram into a CHAIN diagram. This requires a double edge departing from every node. The edges labeled by m_3 and m_4 can be converted into double edges immediately since we have

$$\varphi_3 \rightarrow En(m_3) \qquad \text{and} \qquad \varphi_2 \rightarrow En(m_4)$$

However, $\varphi_1 \not\rightarrow En(\ell_3)$, so we have to do some more work on φ_1.

Example (Cont'd)

The problem with

$$\varphi_1: \ (at\_m_{0..2,5} \vee (at\_m_3 \wedge s = 2)) \wedge at\_\ell_3$$

is the disjunct at_m_5, because

$$at\_m_5 \rightarrow \neg En(\ell_3)$$

Therefore we separate this disjunct and create two new assertions

$$\varphi_1': \quad at\_m_5 \wedge at\_\ell_3$$

$$\varphi_1'': \quad (at\_m_{0..2} \vee (at\_m_3 \wedge s = 2)) \wedge at\_\ell_3$$

As helpful transition for φ_1' we identify m_5. Clearly

$$\varphi_1' \rightarrow En(m_5)$$

and m_5 leads from φ_1' to φ_1''. Now we have

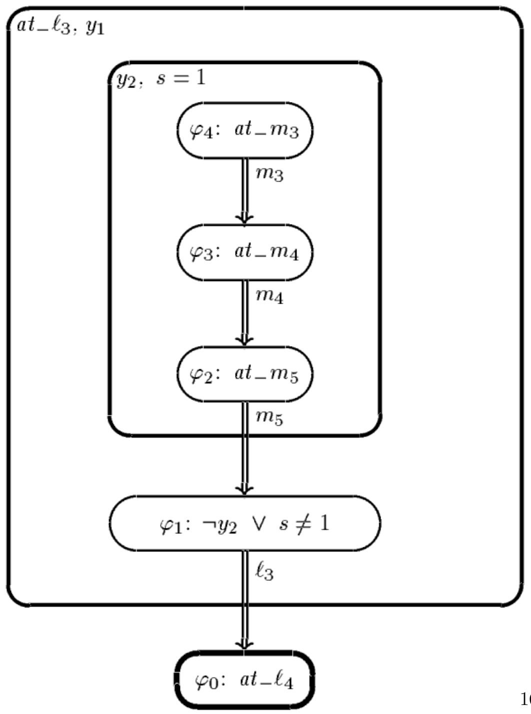
$$\varphi_1'' \rightarrow En(\ell_3)$$

and ℓ_3 leads from φ_1' to φ_0, as required.

With some rearrangement of assertion numbers, and simplification of φ_1'', this leads to the following chain diagram.

Chain diagram for program MUX-PET1

$at\_\ell_3 \Rightarrow \diamondsuit\, at\_\ell_4$



`Example` (Cont'd)

In practice one would not construct a deductive proof like this to prove accessibility (or any property) of MUX-PET1:

MUX-PET1 is a finite-state program (due to the invariant $\chi_1 : s = 1 \lor s = 2$) and therefore fully automatic algorithmic methods are available.

However, the proof by verification diagram does give insight in why the property holds and the possible flows of the program to reach the goal.

Ranking functions: Motivation

In the CHAIN-J rule we used the index of the intermediate assertions as a measure of the distance from the goal. From an intermediate assertion φ_n it takes at most n helpful transitions to reach the goal.

We can generalize this idea of measuring the distance from the goal and define a distance function on the state space, and require that helpful transitions reduce the distance and all other transitions do not increase the distance. This ensures that the goal will eventually be reached.

We will measure the distance with ranking functions which map states into a well-founded domain.

Well-founded domains

Well-founded domain

$$(A, \prec)$$

where A is a set and
$\prec$ is a well-founded order
i.e., there does not exist an infinitely descending sequence $a_0 \succ a_1 \succ a_2 \dots$

Note: A well-founded order is transitive and irreflexive.

Examples:

$(\mathbb{N}, <)$ is well-founded:
$n > n{-}1 > n{-}2 > \dots > 0$

$(\mathbb{Z}, <)$ is not well-founded:
$n > n{-}1 > \dots > 0 > -1 > -2 \dots$

$(\mathbb{Z}, |{<}|)$ with $x\ |{>}|\ y$ iff $|x| > |y|$ is well-founded:
$-7\ |{>}|\ -3\ |{>}|\ 2\ |{>}|\ -1\ |{>}|\ 0$

(Rationals in $[0,1], <)$ is not well-founded:
$1 > \frac{1}{2} > \frac{1}{4} > \frac{1}{8} > \frac{1}{16} > \dots$

Lexicographic Product

Well-founded domains $(A_1, \prec_1)$ and $(A_2, \prec_2)$ can be combined into their

lexicographic product $(A_1 \times A_2, \prec)$

where

$$(a_1, a_2) \prec (b_1, b_2) \quad a_i, b_i \in A_i$$

iff

$$a_1 \prec_1 b_1 \text{ or } (a_1 = b_1 \text{ and } a_2 \prec_2 b_2).$$

$(A_1 \times A_2, \prec)$ is also a well-founded domain.

In general, well-founded domains

$$(A_1, \prec_1), \ldots, (A_n, \prec_n)$$

can be combined into their lexicographic product $(A_1 \times \cdots \times A_n, \prec)$ where

$$(a_1, \ldots, a_n) \prec (b_1, \ldots, b_n) \qquad a_i, b_i \in A_i$$

iff for some j, $1 \leq j \leq n$,

$$a_1 = b_1, \ldots, a_{j-1} = b_{j-1}, \; a_j \prec_j b_j$$

$(A_1 \times \cdots \times A_n, \prec)$ is also a well-founded domain.

Well-founded rule (Motivation)

Consider program N:

in N: **integer where** $N > 0$
local i: **integer**

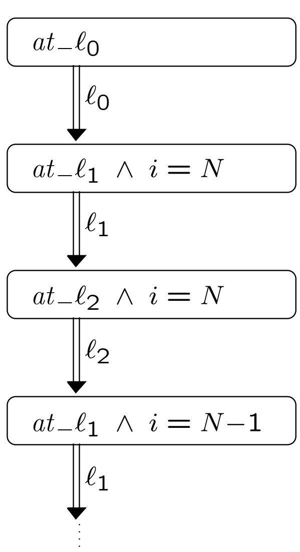
ℓ_0 : $i := N$
ℓ_1 : **while** $i > 0$ **do**
  $\ell_2 : i = i - 1$
ℓ_3 :

We want to prove that for program N:

$$\boxed{at\_\ell_0 \Rightarrow \Diamond\, at\_\ell_3}$$

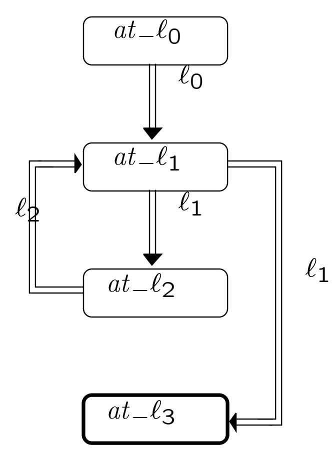
Motivation (Cont'd)

Using CHAIN diagrams to prove this, we would need a separate diagram for each value of N:



which does not seem practical.

What we would like is something like the following diagram:



The problem with this diagram is that it is not acyclic in $\Longrightarrow$.

So how can we be sure that it will eventually exit the cycle to reach the goal?

Rule WELL-J

For assertions $p,\ q = \varphi_0$ and $\varphi_1,\ \ldots,\ \varphi_m$,
helpful transitions $\tau_{h_1},\ \ldots\ \tau_{h_m}\ \in\ \mathcal{J}$,
a well-founded domain $(\mathcal{A}, \prec)$, and
ranking functions $\delta_0\ ,\ldots,\ \delta_m : \Sigma \to \mathcal{A}$

$$
\left.
\begin{array}{ll}
\text{JW1.} & p\ \rightarrow\ \bigvee_{j=0}^{m} \varphi_j \\[1em]
\text{JW2.} & \rho_\tau\ \wedge\ \varphi_i\ \rightarrow\ \left[\begin{array}{l} \bigvee_{j=0}^{m}(\varphi'_j\ \wedge\ \delta_i \succ \delta'_j) \\[1em] \vee(\varphi'_i\ \wedge\ \delta_i = \delta'_i) \end{array}\right] \\
 & \qquad\qquad\qquad \text{for every } \tau \in \mathcal{T} \\[1em]
\text{JW3.} & \rho_{\tau_{h_i}}\ \wedge\ \varphi_i\ \rightarrow\ \bigvee_{j=0}^{m}(\varphi'_j\ \wedge\ \delta_i \succ \delta'_j) \\[1em]
\text{JW4.} & \varphi_i\ \rightarrow\ En(\tau_{h_i})
\end{array}
\right\}(*)
$$

$$p\ \Rightarrow\ \diamondsuit\, q$$

$(*)$ for $i = 1, \ldots, m$

Premise JW2:

In the CHAIN rule we required that all transitions resulted in a move down to a lower-ranked assertion or stay in the same assertion.
Progress towards the goal was measured by the assertion index.

Here, progress is measured by the value of the ranking function, so if a transition reduces the ranking function it may go to any assertion. If it cannot reduce the ranking function it should stay in the same assertion to keep the identity of the helpful transition.

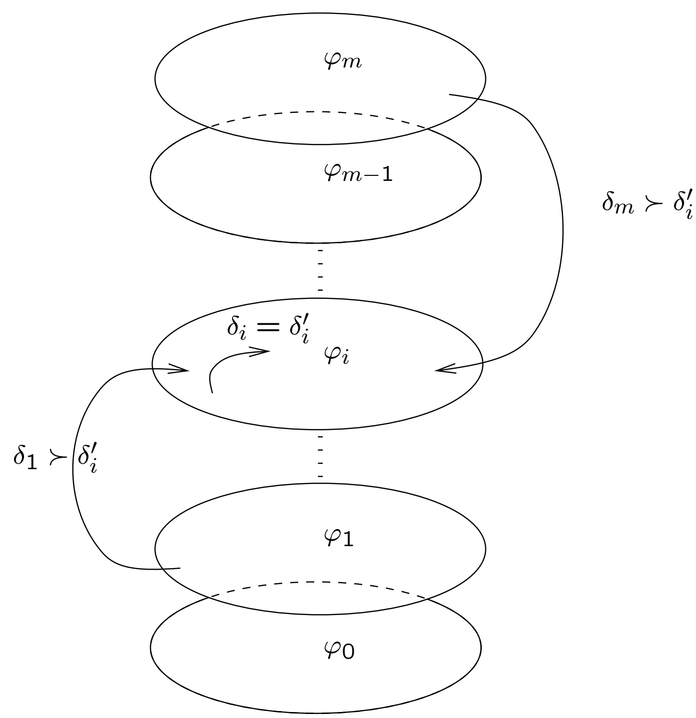
Premise JW3:

The helpful transition is required to reduce the ranking function.

Premise JW4:

Same as in the CHAIN-J rule. It ensures that the helpful transition will eventually be taken, by the justice requirement.

Since $(A, \prec)$ is well-founded there can only be a finite number of those steps, ensuring that eventually φ_0 is reached.



RANK diagrams

Nodes: labeled by assertions and ranking functions

φ_i, δ_i

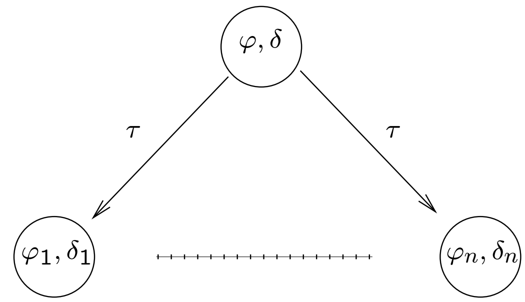
Terminal Nodes:

φ_0, δ_0

Well-formedness constraint:

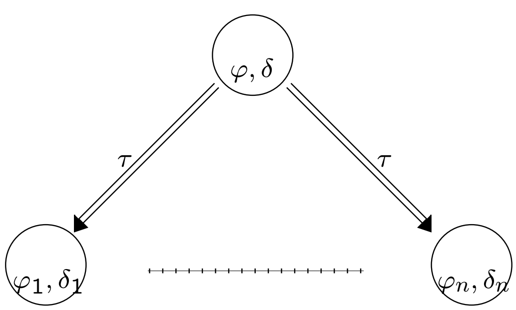
- Every nonterminal node $\varphi_i,\ i > 0$, has a double edge departing from it.
- No transition can label both a single and a double edge departing from the same node.

RANK diagrams: Verification conditions



$$\{\varphi \ \wedge\ \delta = u\} \quad \tau \quad \{(\varphi \wedge u \succeq \delta) \ \vee\ (\varphi_1 \ \wedge\ u \succ \delta_1) \\ \vee\ \ldots\ \vee (\varphi_n \ \wedge\ u \succ \delta_n)\}$$

Verification conditions (Cont'd)



$$\{\varphi \ \wedge\ \delta = u\} \quad \tau \quad \{(\varphi_1 \ \wedge\ u \succ \delta_1) \\ \vee\ \ldots\ \vee\ (\varphi_n \ \wedge\ u \succ \delta_n)\}$$

$\varphi \rightarrow En(\tau)$

Claim: A P-valid rank diagram establishes that

$$\bigvee_{j=0}^{m} \varphi_j \Rightarrow \Diamond\, \varphi_0$$

is P-valid.

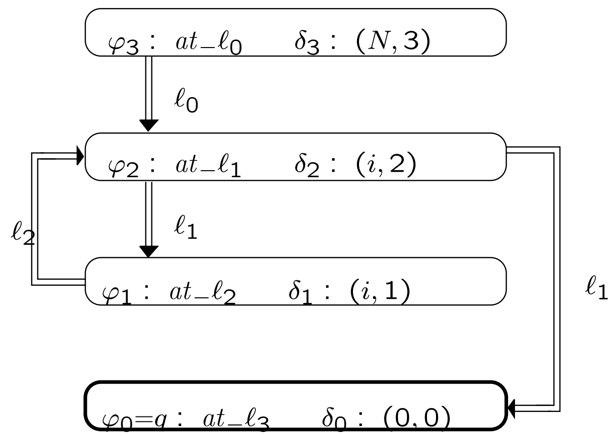
With $\quad p \rightarrow \bigvee_{j=0}^{m} \varphi_j \quad$ and $\quad \varphi_0 \rightarrow q$,

we can conclude the P-validity of

$$\boxed{p \Rightarrow \Diamond\, q}$$

Example: Program N

Verification diagram for program N
and property
$\underbrace{at\_\ell_0}_{p} \Rightarrow \Diamond \underbrace{at\_\ell_3}_{q}$



Example (Cont'd): Verification conditions

- $\underbrace{at\_\ell_0}_{p} \ \rightarrow \ \underbrace{at\_\ell_0}_{\varphi_3} \vee \varphi_2 \ \vee \ \varphi_1 \ \vee \ \varphi_0$

Four double lines:

- $\varphi_1 \Rightarrow \varphi_2$:

$$\underbrace{at\_\ell_2 \wedge at'\_\ell_1 \wedge i' = i - 1}_{\rho_{\ell_2}} \wedge \ldots$$
$$\ldots \wedge \underbrace{at\_\ell_2}_{\varphi_1} \wedge u = \underbrace{(i, 1)}_{\delta_1} \ \rightarrow$$
$$\underbrace{at'\_\ell_1}_{\varphi'_2} \wedge (\underbrace{(i, 1)}_{\delta_1} \succ \underbrace{(i', 2)}_{\delta'_2})$$

- $\underbrace{at\_\ell_2}_{\varphi_1} \ \rightarrow \ \underbrace{at\_\ell_2}_{En(\ell_2)}$

Example: Program INC

local y, inc : **integer where** $y \geq 0 \wedge inc = 1$

$$\left[\begin{array}{l} \ell_0: \ \textbf{while } y > 0 \textbf{ do} \\ \qquad \ell_1: \ y := y + inc \\ \ell_2: \end{array}\right] \ || \ \left[\begin{array}{l} m_0: \ inc := 0 \\ m_1: \ inc := -1 \\ m_2: \end{array}\right]$$

We want to prove for program INC

$$\boxed{at\_\ell_0 \Rightarrow \Diamond \ at\_\ell_2}$$

Invariants:

$$at\_m_0 \ \rightarrow \ inc = 1$$
$$at\_m_1 \ \rightarrow \ inc = 0$$
$$at\_m_2 \ \rightarrow \ inc = -1$$

While at m_0 and at m_1 no progress is made by traversing the loop ℓ_0–ℓ_1. Progress is made only by moving to m_2.

While at m_2, progress is made by executing ℓ_0 and ℓ_1, so the loop is made explicit in the diagram.

RANK diagram for program INC
representing the proof of
$at\_\ell_0 \Rightarrow \Diamond \ at\_\ell_2$