Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs

Bertrand JEANNET and $\boxed{\text{Wendelin SERWE}}$

MODOCOP

IRISA
institut de recherche en informatique
et systèmes aléatoires

INRIA

Aim

analysis of programs with:

- recursive *procedures*
- *local* variables with *infinite* domains (e.g., $\mathbb{N}$, $\mathbb{R}$)
- *global* variables with *infinite* domains (e.g., $\mathbb{N}$, $\mathbb{R}$)

using *abstract interpretation*

Intraprocedural Analysis

Program States: $\mathit{Ctrl} \times \mathbb{N}^l \times \mathbb{N}^g$

control points of the program

values of global variables

values of local variables

- known techniques for abstract interpretation:
  - intervals [CousotCousot76]
  - polyhedrons [CousotHalbwachs78]
  - octagons [Miné00]

    ⋮
- applications:
  - elimination of run-time errors [BCCFMMMR02]
  - verification of reactive systems [Halbwachs93, HJR99]

    ⋮

Interprocedural Analysis

Program States: $\left(\mathit{Ctrl} \times \mathbb{N}^l\right)^* \times \mathbb{N}^g$

control points of the program values of global variables

values of local variables

some possibilities:

- abstract the stack: keep only the lop local environment
  problem: how to restore the calling environment upon return?

- fix the depth of the stack:
  problem: imprecision for programs exceeding this depth

our approach:

- keep possible calling environments

- select matching environments on procedure returns
  ($\Rightarrow$ call by value, no modification of formal parameters)

Related Work

Predicate Transformers [CousotCousot77, CousotHalbwachs78]:

- based on a denotational semantics
- no global variables (we have the same precision in this case)

[Bourdoncle90]:

- more complex program model: nested procedures, different passing schemes, aliasing variables on the stack, . . .
- somewhat complicated
  (⇒ implemented only for non-relational domains)

using Push Down Systems [EsparzaKnoop99, BouajjaniEsparzaTouili03]:

- abstract all data types *finitely*
- represent call stacks by regular languages

Plan

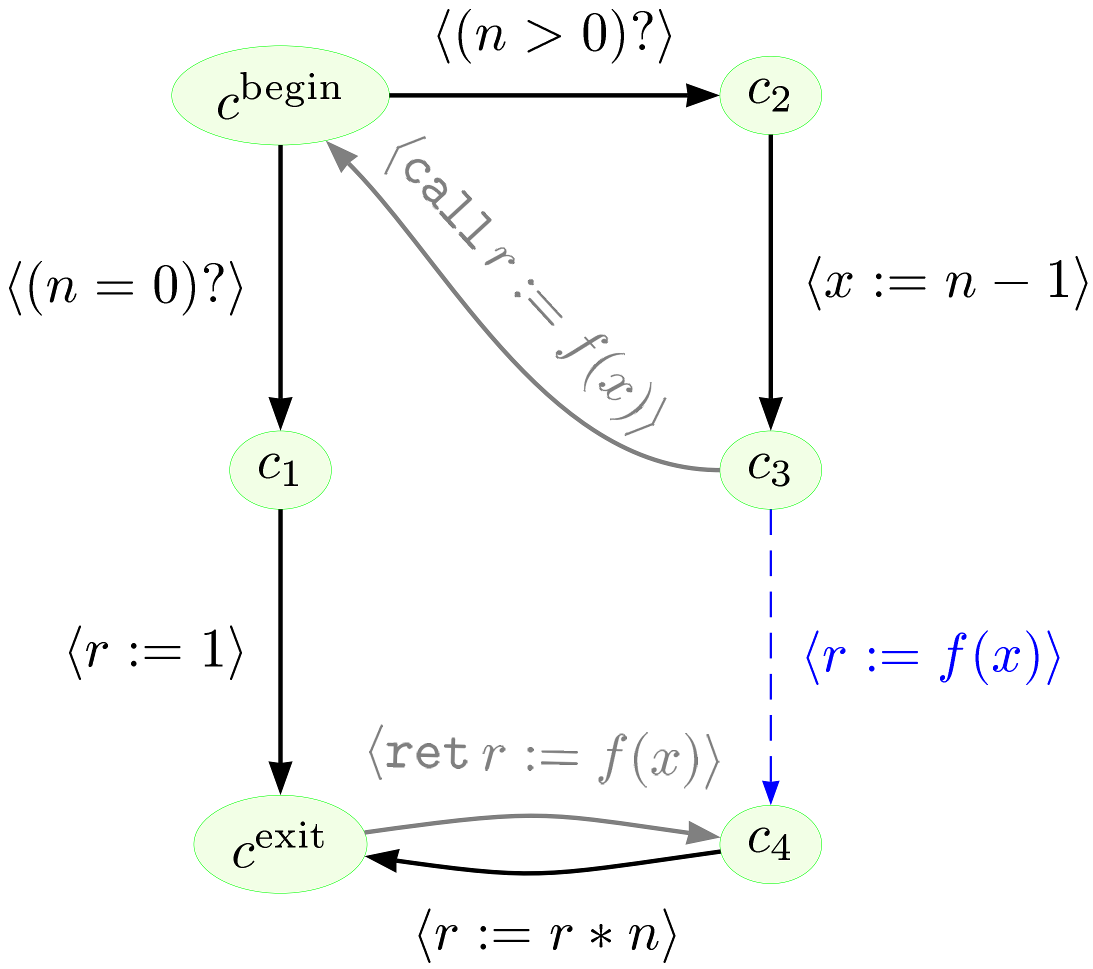
1. Introduction
2. Related Work
3. Program Model: Syntax, Semantics & Properties
4. A More Abstract Semantics: Definition & Properties
5. Abstract Interpretation
6. Conclusion

Program Model – informally

- imperative programs with procedures
- boolean, enumerated and numerical variables
- fixed set of global and local variables (no heap)
- call-by-value with no modification of input parameters
- return parameters
- no nested procedures

starting point: interprocedural control flow graph

Program Model – Example: Factorial



procedure f (factorial)

- local variables: $\{n, x, r\}$
- formal parameter: n
- return parameter: r
- control points: $\{c^{\text{begin}}, c^{\text{exit}}, c_1, c_2, c_3, c_4\}$

Program Model – formally – 1

Program $\langle GVar, (P_i)_{0\leq i\leq p}\rangle$

- global variables: $GVar$
- procedures: $(P_i)_{0\leq i\leq p}$
- main procedure: P_0

Procedure $\langle LVar_i, FPar_i, FRet_i, G_i\rangle$

- local variables: $LVar_i$
- formal parameters: $\langle \mathbf{fp}_i^1, \ldots, \mathbf{fp}_i^m\rangle = FPar_i \subseteq LVar_i$
- return parameters: $\langle \mathbf{fr}_i^1, \ldots, \mathbf{fr}_i^n\rangle = FRet_i \subseteq LVar_i$
- intra-procedural flow graph: G_i

Program Model – formally – 2

Intra-procedural flow graph (for P_i) $\langle \mathit{Ctrl}_i, I_i \rangle$

- control points: Ctrl_i
- *single* entry and return: c_i^{begin}, c_i^{exit}
- instructions: $I_i : \mathit{Ctrl}_i \times \mathit{Ctrl}_i \to I_\perp$
  $I_i(c, c') = \perp$ iff $\nexists$ edge from c to c'

Inter-procedural flow graph $G^* = \langle \bigcup_i \mathit{Ctrl}_i, I^* \rangle$

$$\frac{I_i(c, c') \neq \langle \mathbf{y} := P_j(\mathbf{x}) \rangle}{I^*(c, c') = I_i(c, c')} \qquad \frac{I_i(c, c') = \langle \mathbf{y} := P_j(\mathbf{x}) \rangle}{\begin{array}{rcl} I^*(c, c_j^{\text{begin}}) & = & \langle \mathtt{call}\, \mathbf{y} := P_j(\mathbf{x}) \rangle \\ I^*(c_j^{\text{exit}}, c') & = & \langle \mathtt{ret}\, \mathbf{y} := P_j(\mathbf{x}) \rangle \end{array}}$$

Program Model – formally – 3

Instructions I

- $\langle x := e \rangle$ assignments
- $\langle e? \rangle$ tests
- $\langle \mathbf{y} := P_j(\mathbf{x}) \rangle$ procedure calls, where $\mathbf{x}$ and $\mathbf{y}$ are vectors of variables

Instructions I^*

- $I \setminus \big\{ \langle \mathbf{x} := P_j(\mathbf{x}) \rangle \big\}$
- $\langle \mathtt{call}\, \mathbf{y} := P_j(\mathbf{x}) \rangle$ (procedure) calls
- $\langle \mathtt{ret}\, \mathbf{y} := P_j(\mathbf{x}) \rangle$ (procedure) returns

Operational Semantics

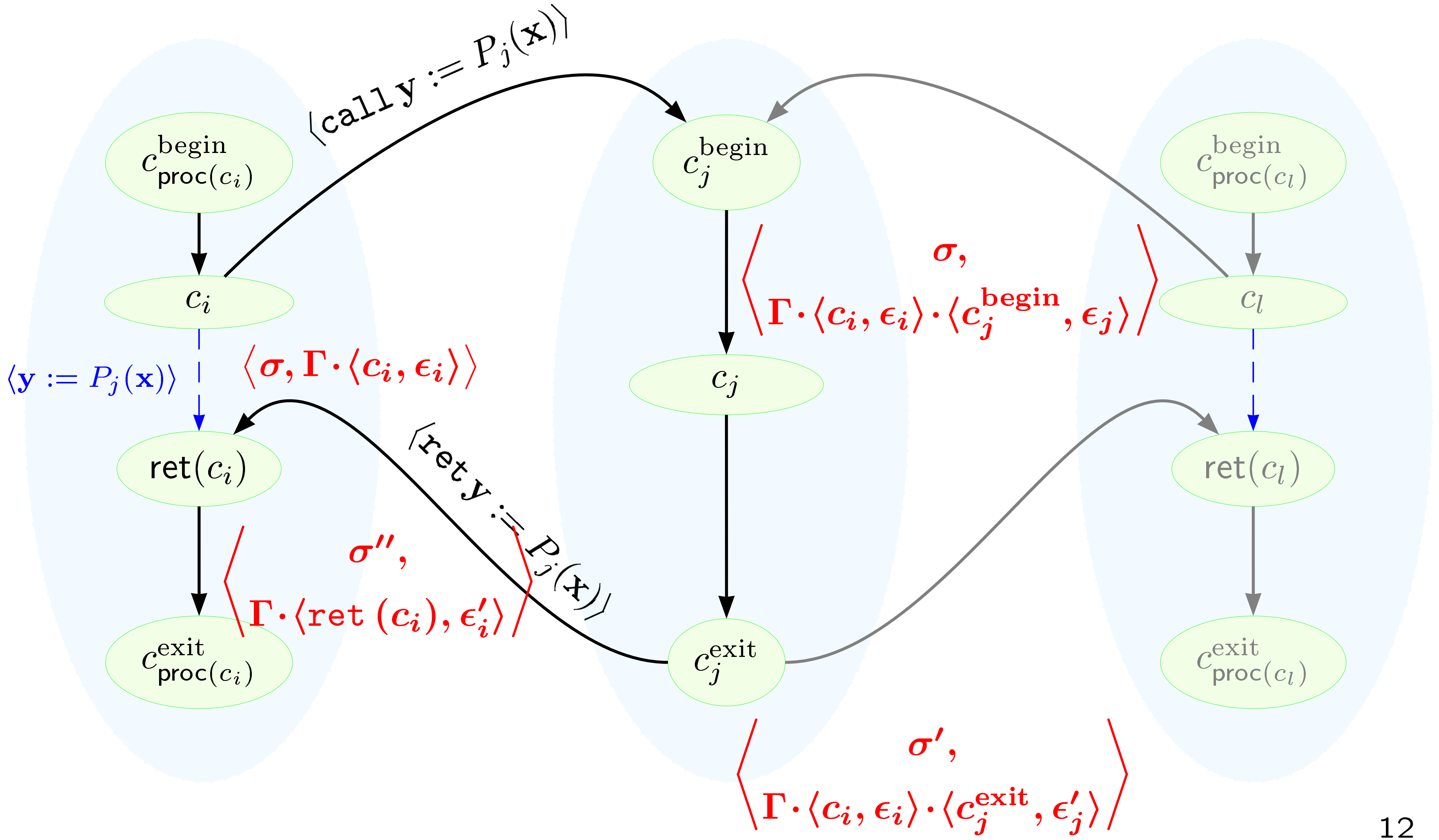
- defined by a transition system $\langle \boldsymbol{State}, \rightarrow \rangle$
- $\boldsymbol{State}$: set of program states

$$\left\langle \boldsymbol{\sigma},\ \boldsymbol{\Gamma} \right\rangle \in \boldsymbol{State}$$

$$\boldsymbol{State} = \boldsymbol{GEnv} \times (\boldsymbol{Ctrl} \times \boldsymbol{LEnv})^{+}$$

  - global environment: $\boldsymbol{\sigma}$
  - stack: $\boldsymbol{\Gamma} = \langle \boldsymbol{c_0}, \boldsymbol{\epsilon_0} \rangle \cdot \ldots \cdot \langle \boldsymbol{c_n}, \boldsymbol{\epsilon_n} \rangle$
    sequence of pairs of control points and local environments
  - current control point: $\boldsymbol{c_n}$
  - local environment: $\boldsymbol{\epsilon_n}$
- $\rightarrow$: transition relation, defined by inference rules

Operational Semantics – Example



Operational Semantics – Inference Rule for `call`

$$\frac{\begin{array}{c} I^*(c, c') = \langle \texttt{call}\, \mathrm{y} := P_j(\mathrm{x}) \rangle \\ \epsilon_j = \left[\mathrm{fp}_j^{(k)} \mapsto (\sigma \oplus \epsilon)\big(\mathrm{x}^{(k)}\big) \right] \end{array}}{\big\langle \sigma, \Gamma \cdot \langle c, \epsilon \rangle \big\rangle \to \big\langle \sigma, \Gamma \cdot \langle c, \epsilon \rangle \cdot \langle c_j^{\mathrm{begin}}, \epsilon_j \rangle \big\rangle}$$

Standard Collecting Semantics

reachable states $reach$

$$\begin{aligned} & \wp(\boldsymbol{State}) \rightarrow \wp(\boldsymbol{State}) \\ & \quad \boldsymbol{X_0} \mapsto \{\boldsymbol{q} \mid \exists \boldsymbol{q}^{\boldsymbol{init}} \in \boldsymbol{X_0} \ . \ \boldsymbol{q}^{\boldsymbol{init}} \rightarrow^* \boldsymbol{q}\} \end{aligned}$$

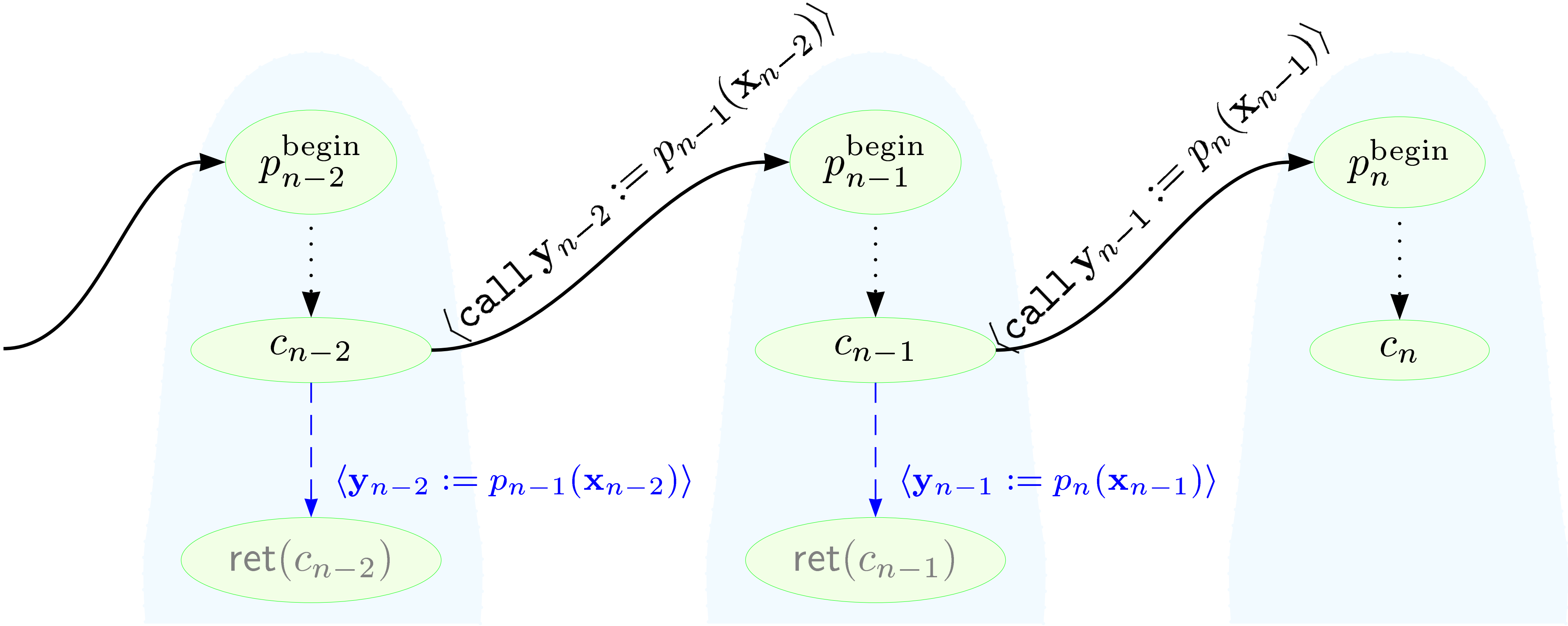
compute reach as the fixpoint of

$$\boldsymbol{X} = \boldsymbol{X_0} \cup \boldsymbol{post}(\boldsymbol{X})$$

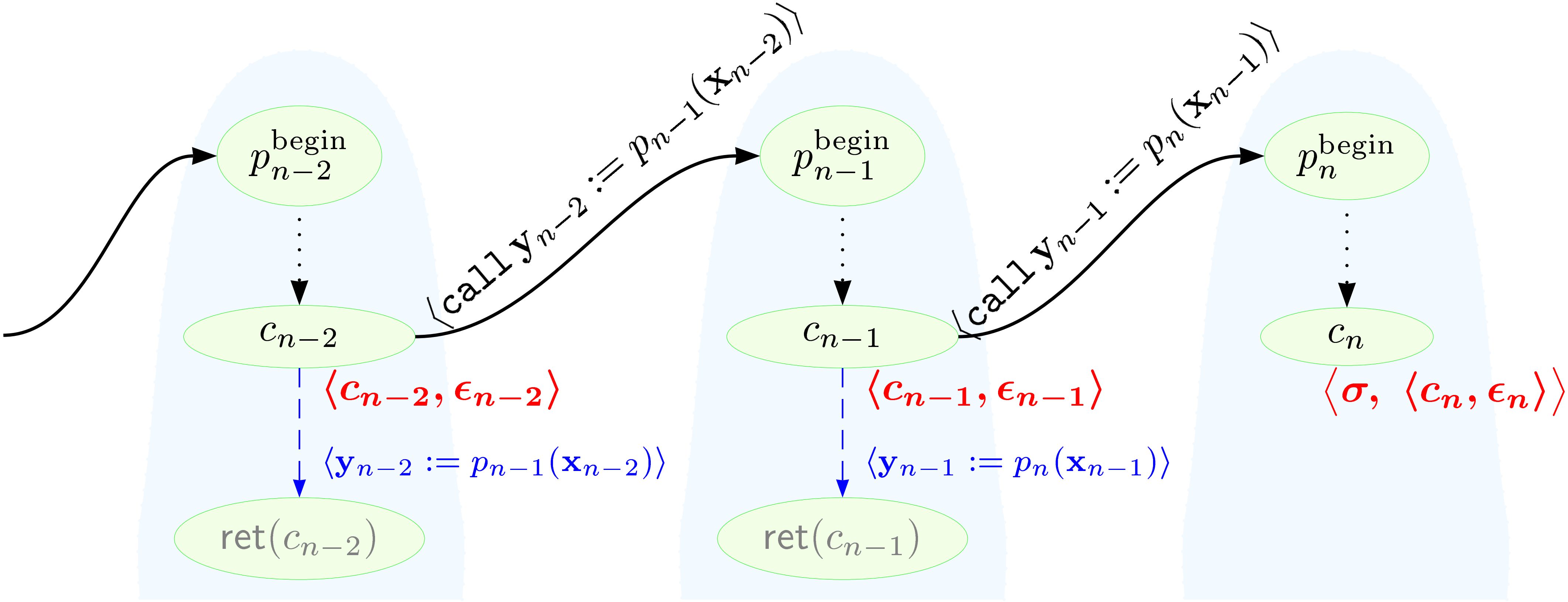
where the forward transfer function $\boldsymbol{post}$ is defined as

$$\begin{aligned} & \wp(\boldsymbol{State}) \rightarrow \wp(\boldsymbol{State}) \\ & \quad \boldsymbol{X} \mapsto \left\{\boldsymbol{q'} \mid \exists \boldsymbol{q} \in \boldsymbol{X} : \boldsymbol{q} \rightarrow \boldsymbol{q'}\right\} = \bigcup_{(\boldsymbol{c},\boldsymbol{c'}) \in \boldsymbol{Ctrl} \times \boldsymbol{Ctrl}} \boldsymbol{post}(\boldsymbol{c} \xrightarrow{\boldsymbol{I}^*(\boldsymbol{c},\boldsymbol{c'})} \boldsymbol{c'})(\boldsymbol{X}) \end{aligned}$$

Op. Semantics – Properties:



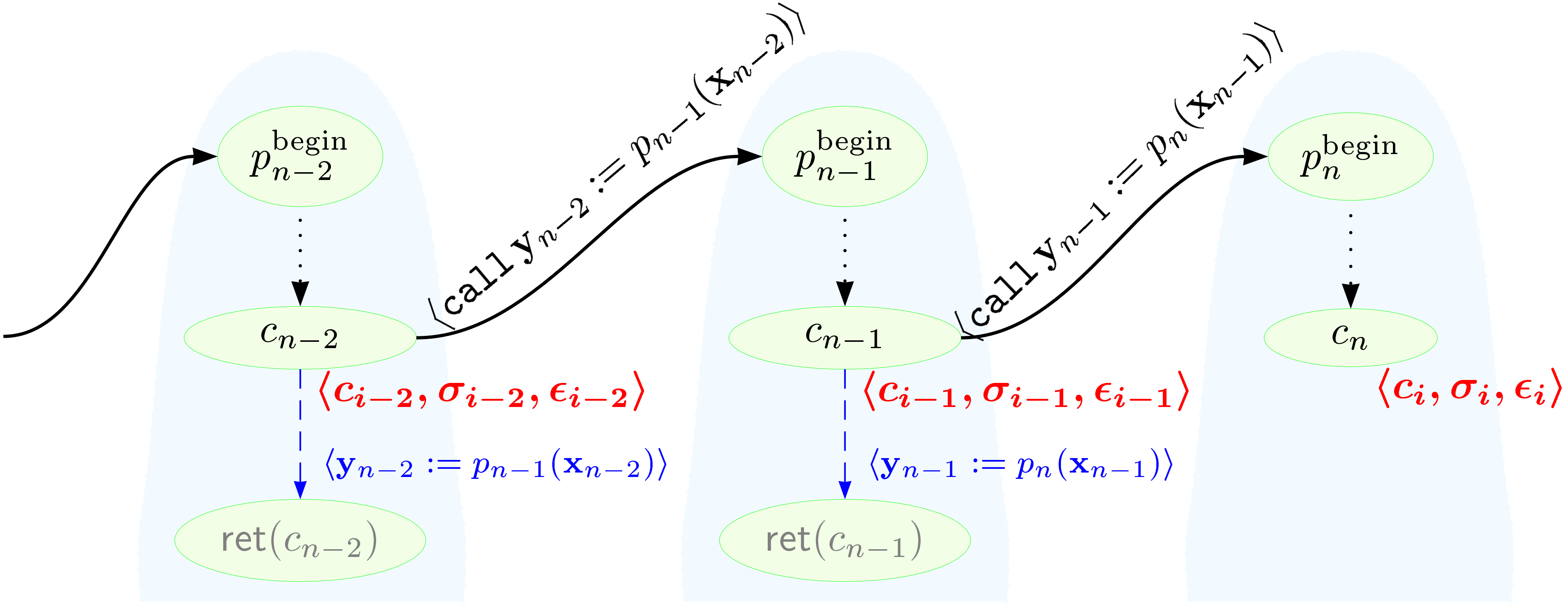
Op. Semantics – Properties: Consistent States



local variables of caller = formal parameters of callee

$$\forall\, 0 < i \leq n,\ \forall \mathrm{x} \notin GVar: \quad \epsilon_{i-1}(\mathrm{x}) = \epsilon_i(\mathrm{fp}_i)$$

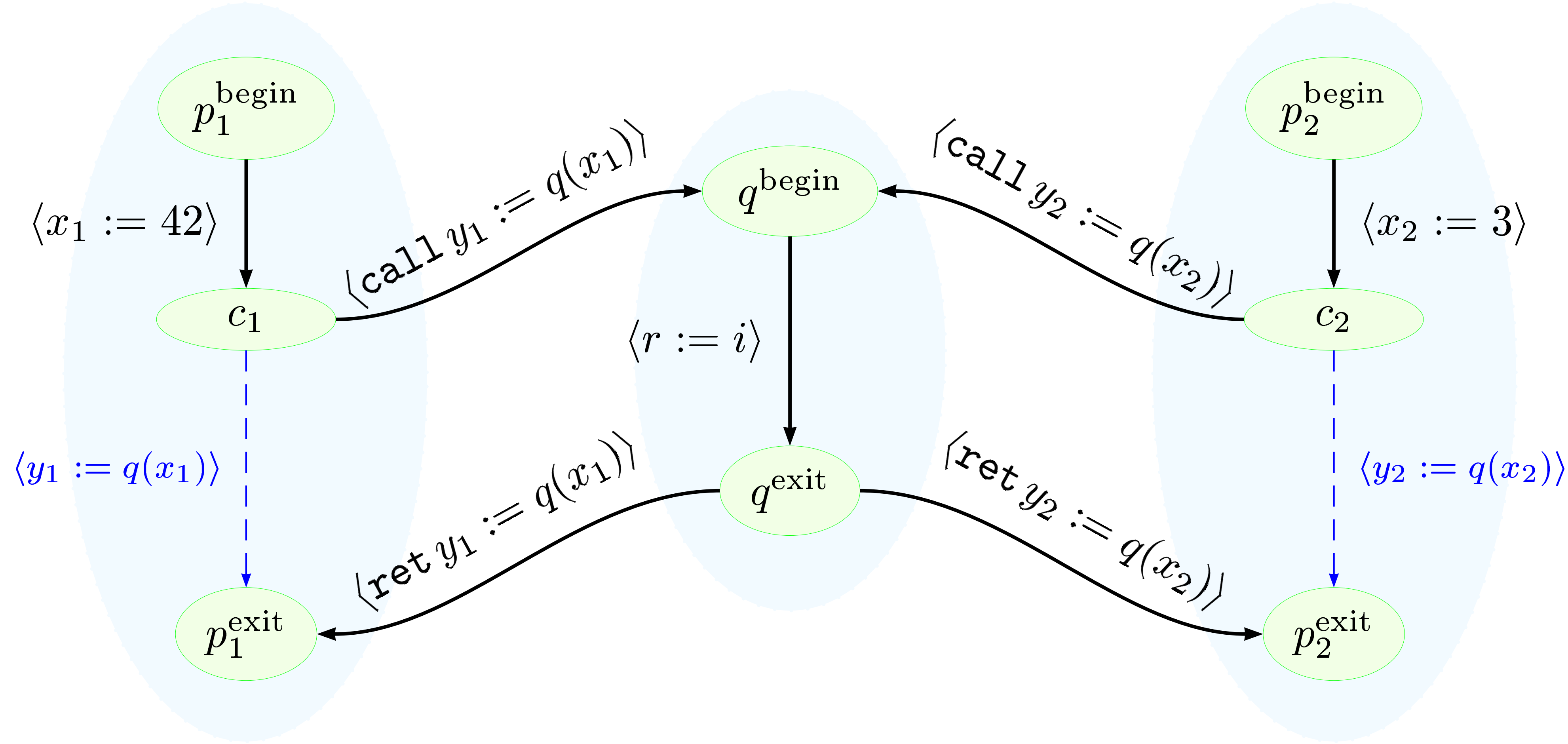
Op. Semantics – Properties: Valid Calling Contexts



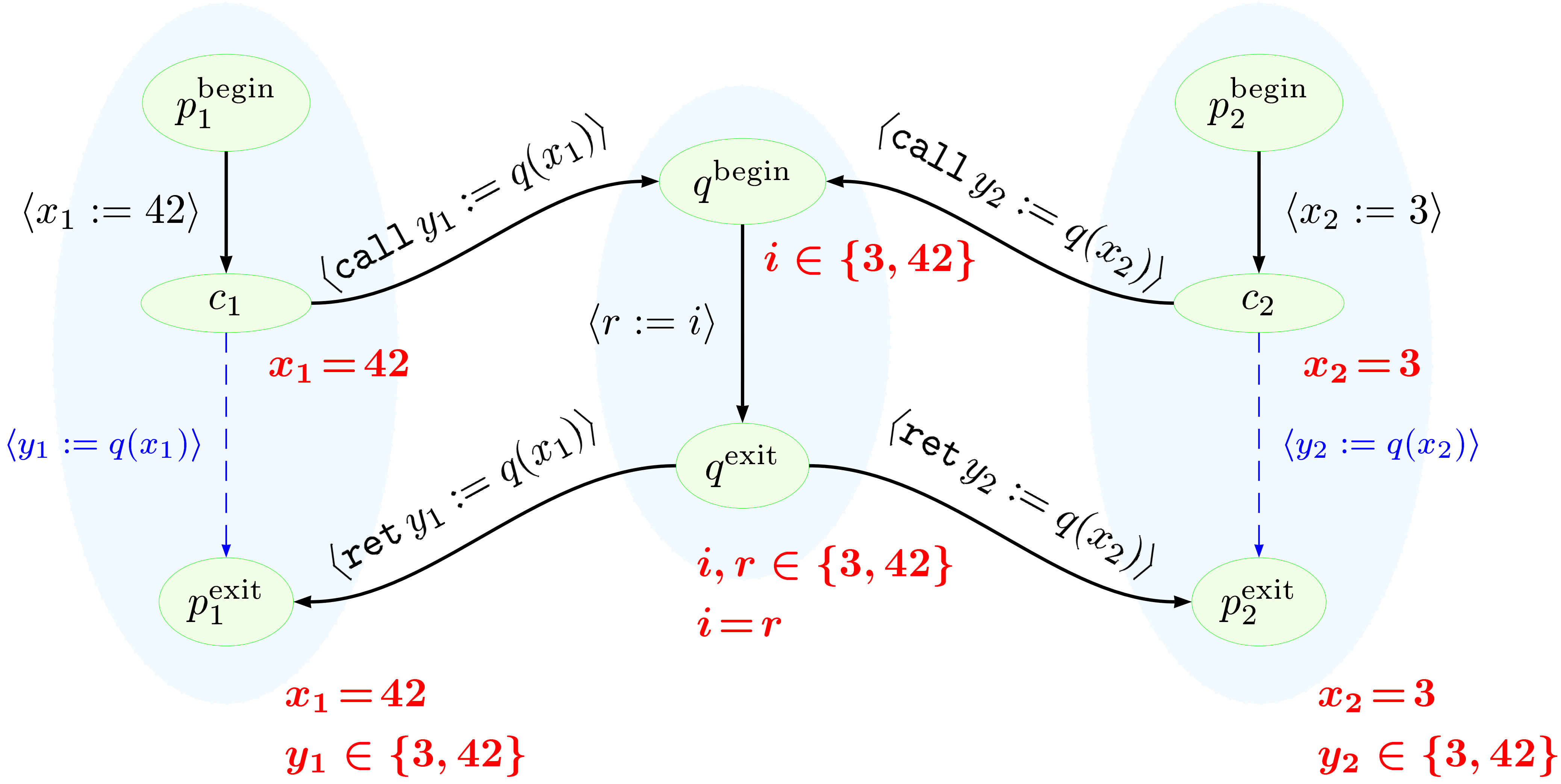
local and global variables of caller = formal parameters of callee

$$\forall\, 0 < i \leq n, \begin{cases} \forall \mathrm{x} \in \mathit{GVar}: & \sigma_{i-1}(\mathrm{x}) = \epsilon_i(\mathrm{fp}_i) \\ \forall \mathrm{x} \notin \mathit{GVar}: & \epsilon_{i-1}(\mathrm{x}) = \epsilon_i(\mathrm{fp}_i) \end{cases}$$

Analysis – Example 1

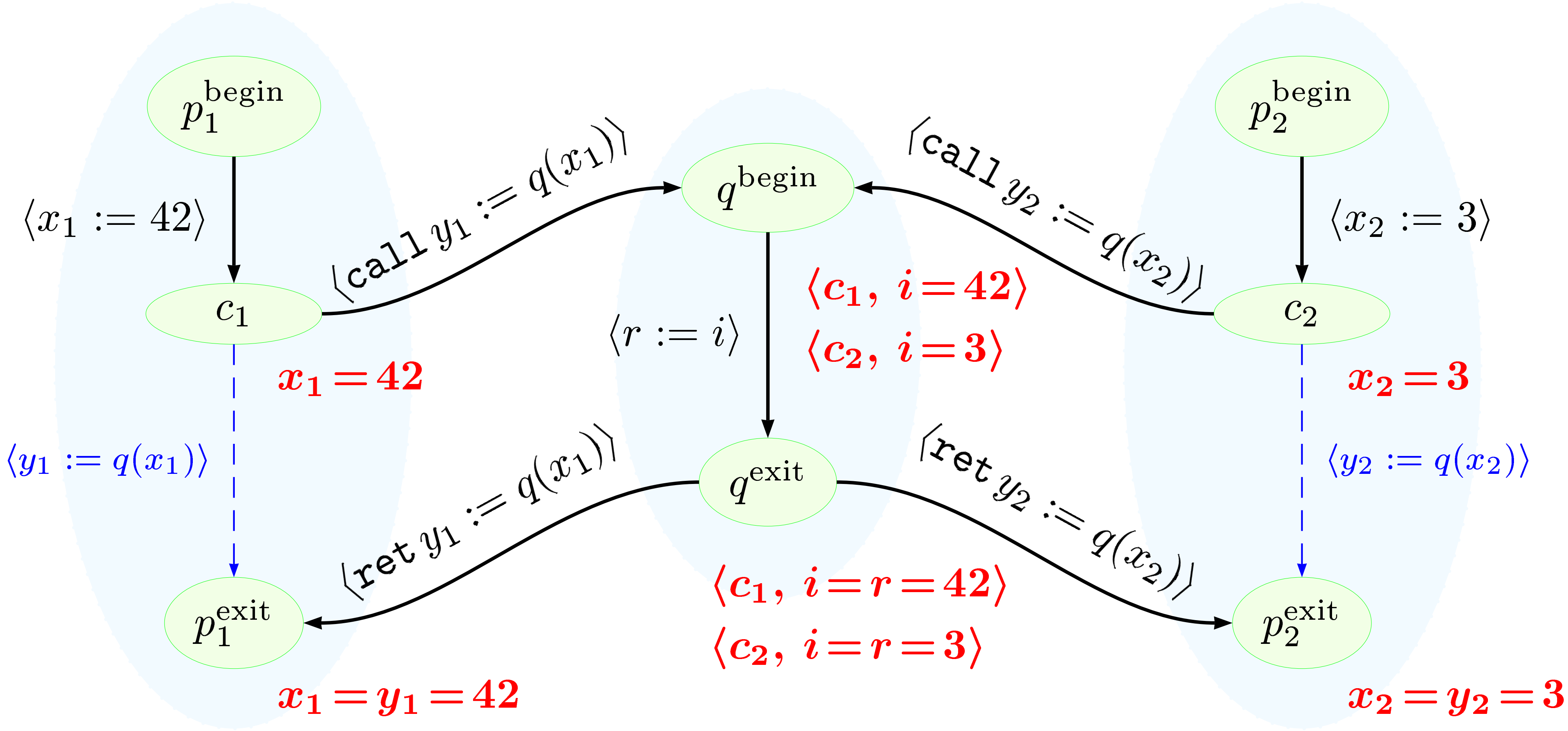


Analysis – Example 1



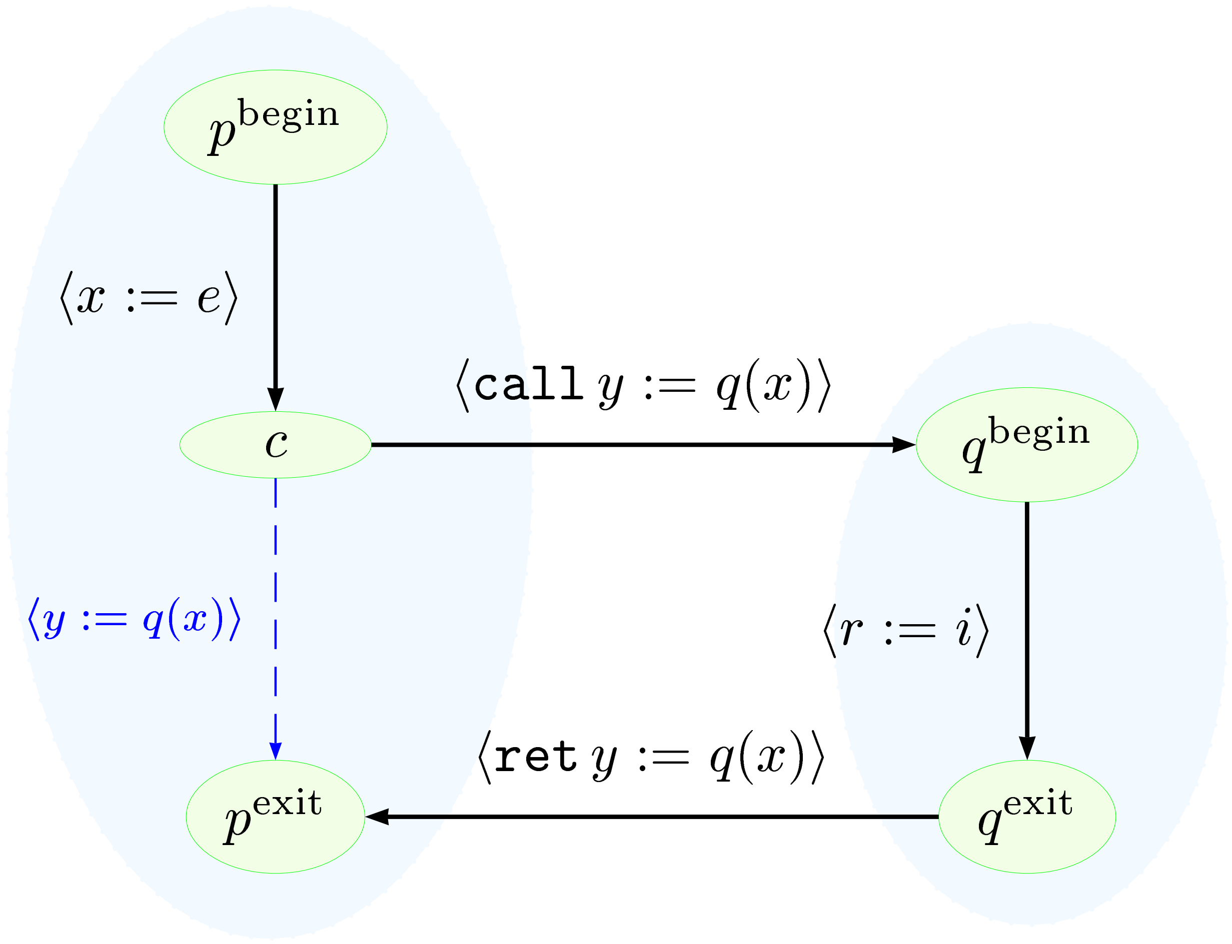
"fusion *all* calls to a procedure"

Analysis – Example 1

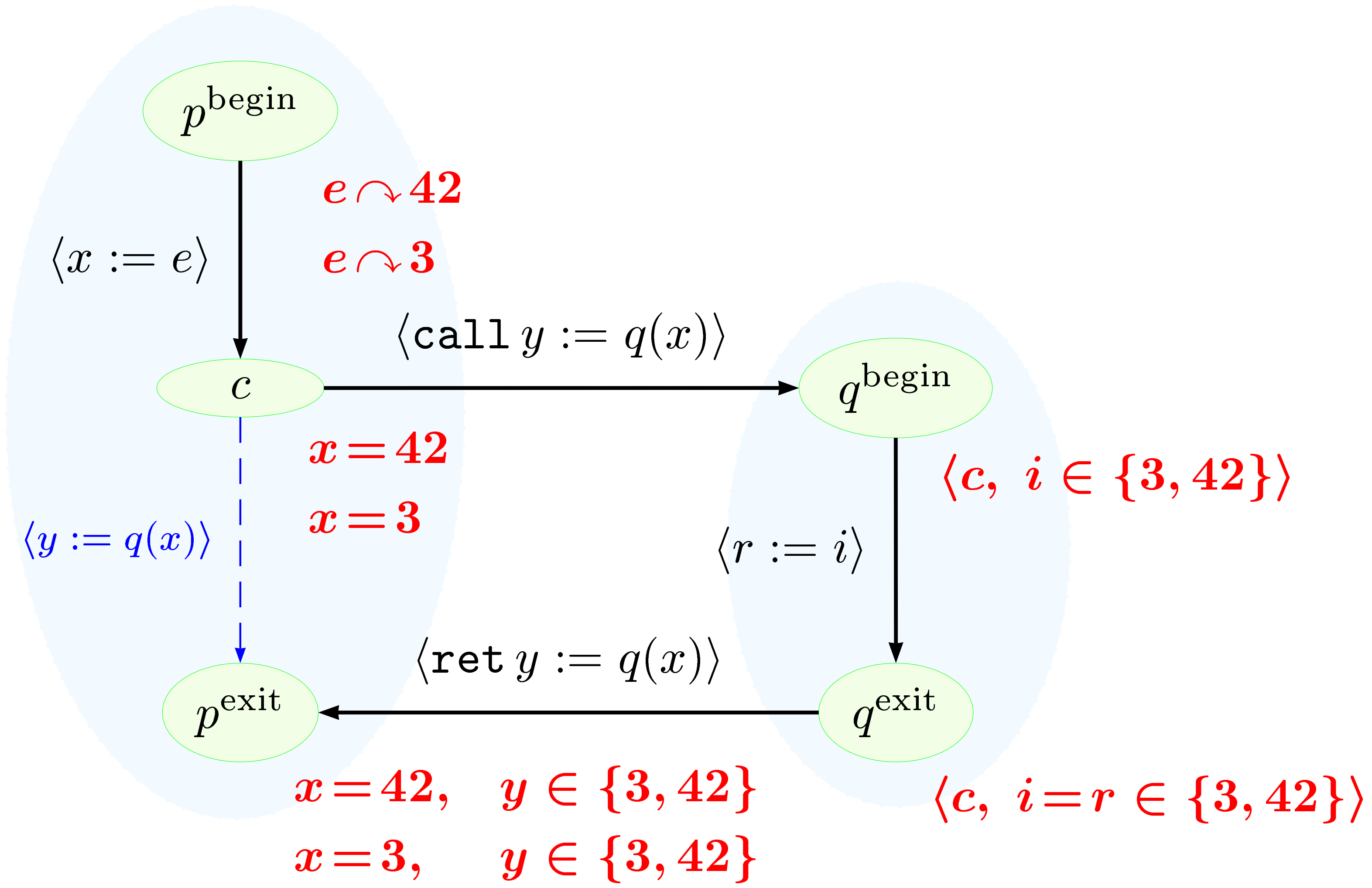


"calls from *different* call-sites can be distinguished"

Analysis – Example 2

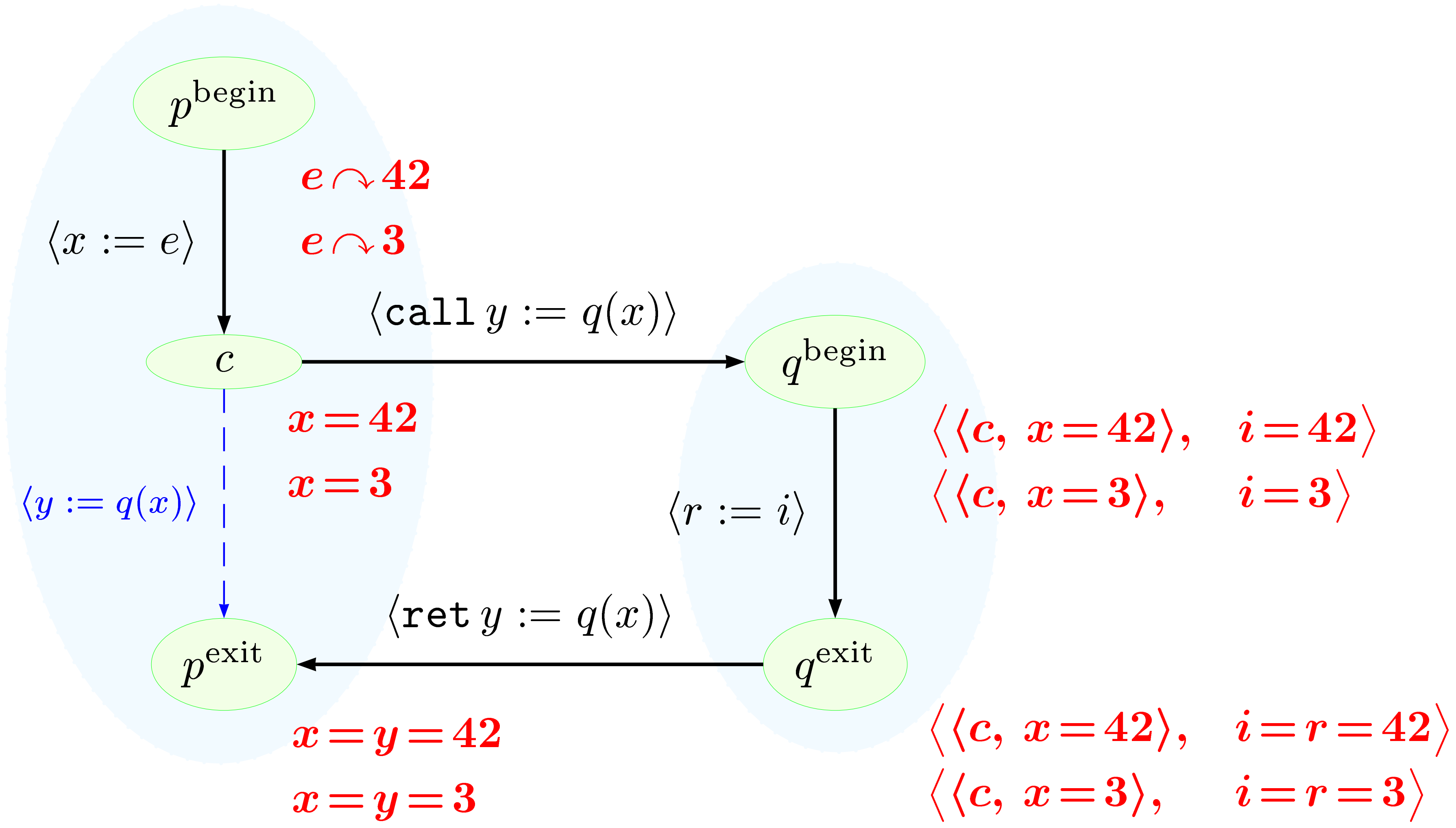


Analysis – Example 2



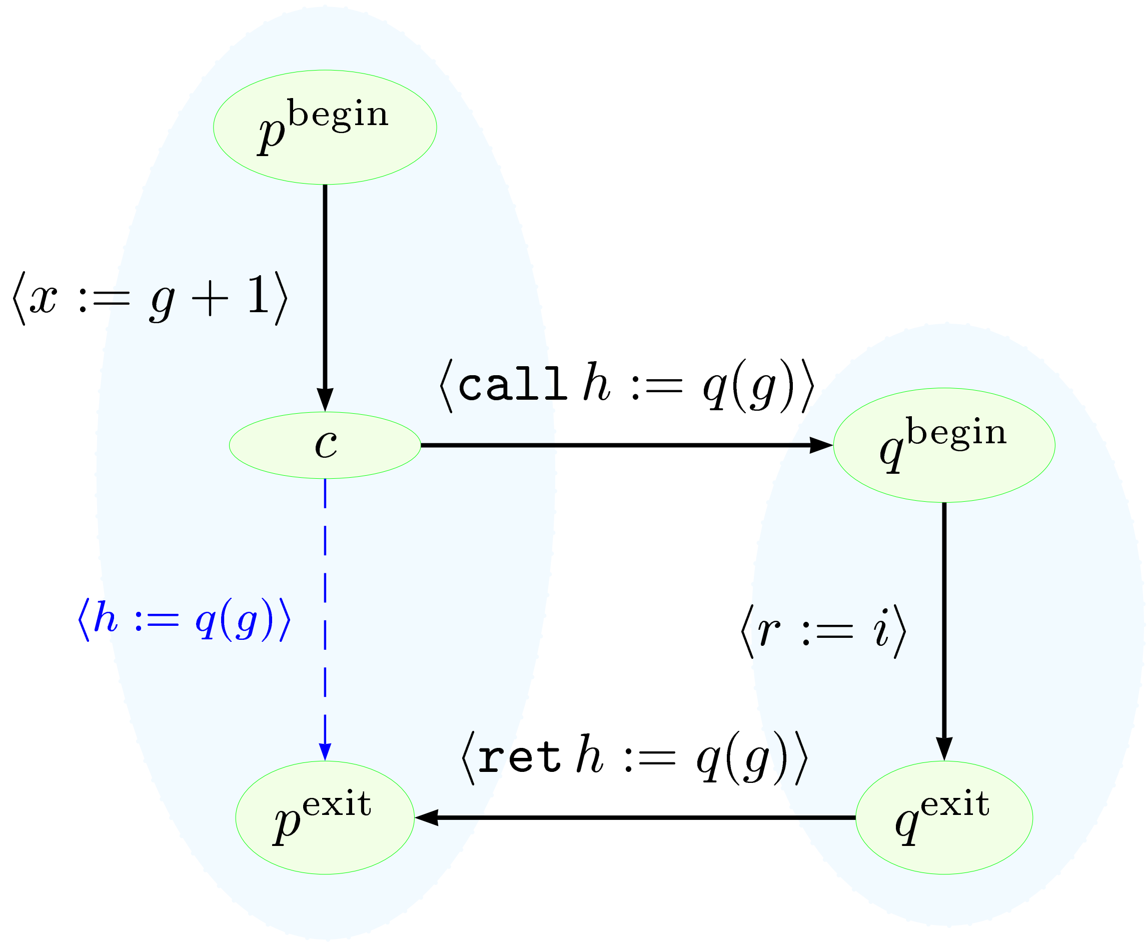
"fusion calls with *different* arguments"

Analysis – Example 2

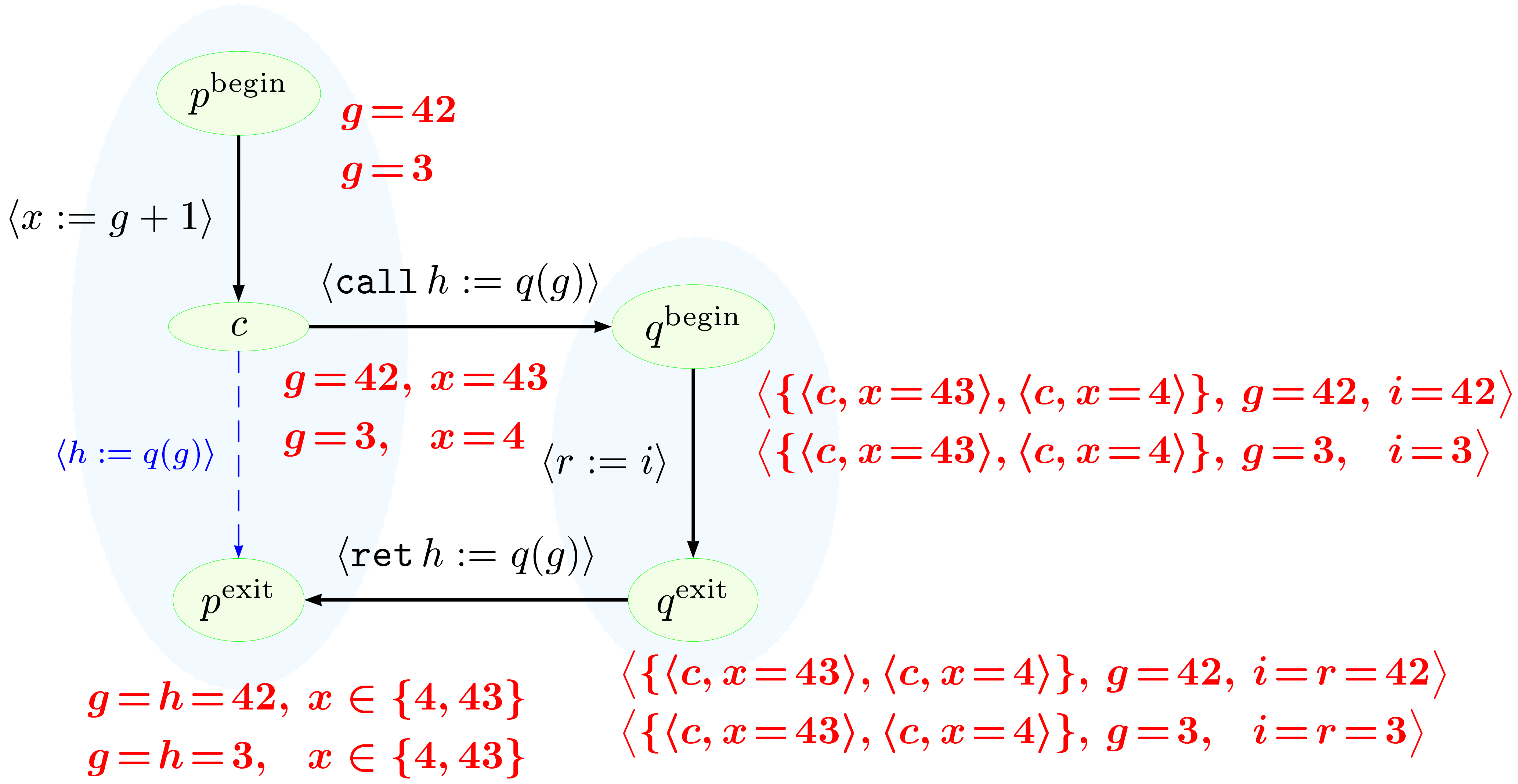


"distinguish calls with *different* local environments"

Analysis – Example 3

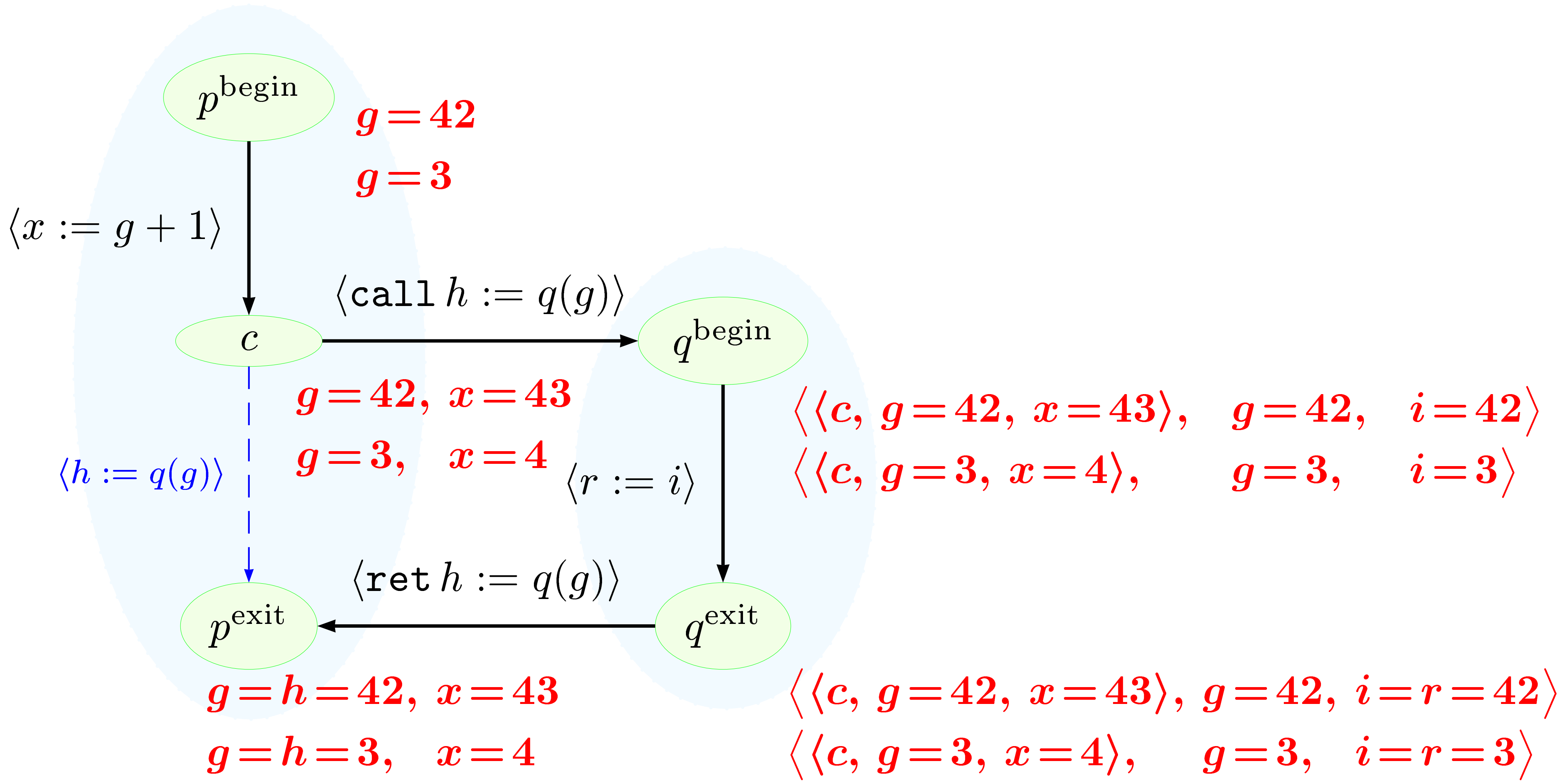


Analysis – Example 3



“fusion calls with *different* global environments

Analysis – Example 3



"distinguish calls with *different* global environments"

Abstract States

$$\boldsymbol{A} \stackrel{\mathbf{def}}{=} \wp(\boldsymbol{Ctrl}^{+} \times \boldsymbol{GEnv} \times \boldsymbol{LEnv}) \times \wp(\boldsymbol{Ctrl}^{+} \times \boldsymbol{GEnv} \times \boldsymbol{LEnv})$$

$\langle \overline{\boldsymbol{Y}}, \widetilde{\boldsymbol{Y}} \rangle \in \boldsymbol{A}$

$\overline{\boldsymbol{Y}}$**:** simplified possible states: keep only the top local environment

$\widetilde{\boldsymbol{Y}}$**:** valid calling contexts: states from which a call might have come

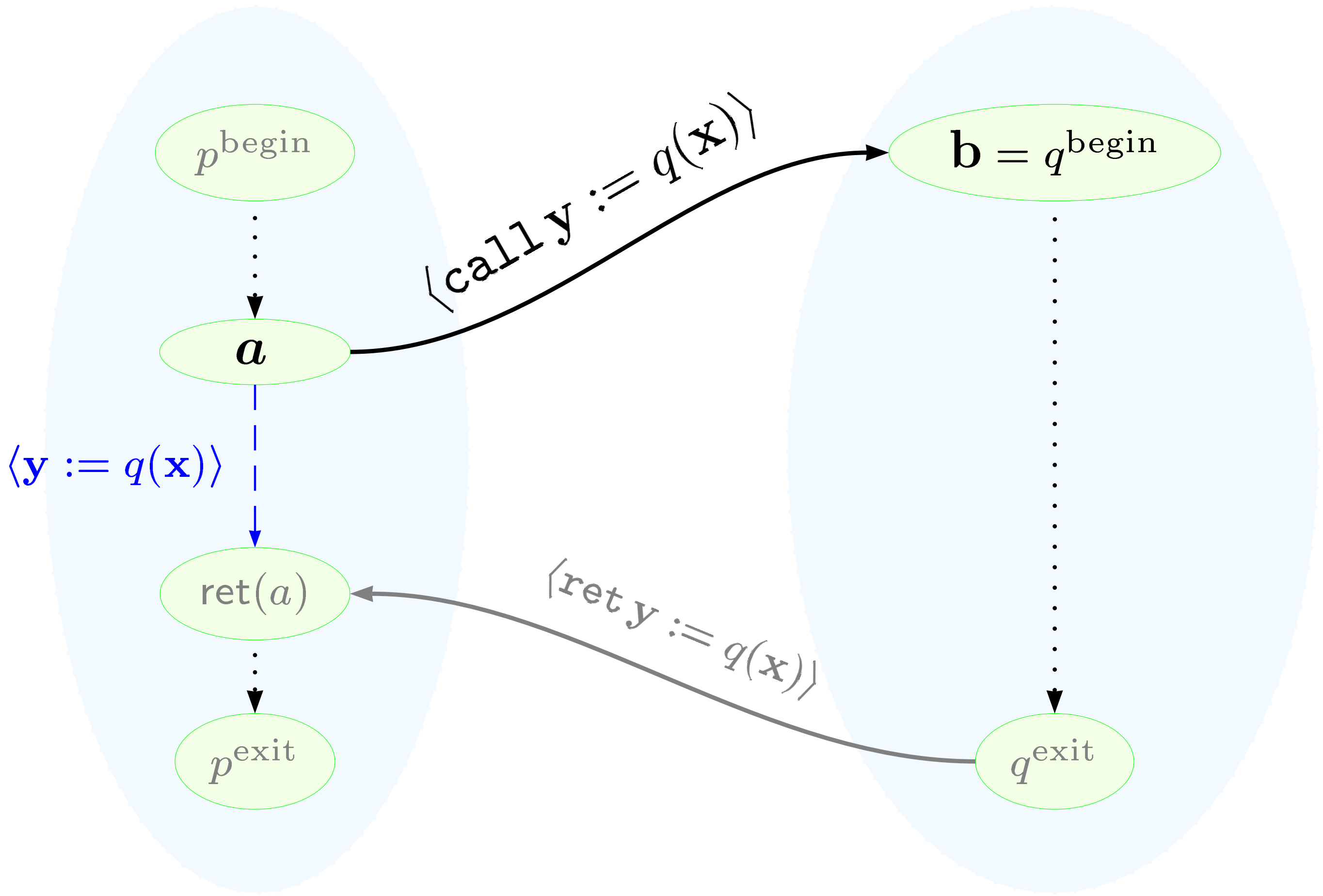
$\langle \boldsymbol{A}, \sqcup, \sqcap, \sqsubseteq \rangle$ is a *lattice*

$$\left\langle \overline{\boldsymbol{A_1}}, \widetilde{\boldsymbol{A_1}} \right\rangle \sqcup \left\langle \overline{\boldsymbol{A_2}}, \widetilde{\boldsymbol{A_2}} \right\rangle \stackrel{\mathbf{def}}{=} \left\langle \overline{\boldsymbol{A_1}} \cup \overline{\boldsymbol{A_2}}, \widetilde{\boldsymbol{A_1}} \cup \widetilde{\boldsymbol{A_2}} \right\rangle$$

$$\left\langle \overline{\boldsymbol{A_1}}, \widetilde{\boldsymbol{A_1}} \right\rangle \sqcap \left\langle \overline{\boldsymbol{A_2}}, \widetilde{\boldsymbol{A_2}} \right\rangle \stackrel{\mathbf{def}}{=} \left\langle \overline{\boldsymbol{A_1}} \cap \overline{\boldsymbol{A_2}}, \widetilde{\boldsymbol{A_1}} \cap \widetilde{\boldsymbol{A_2}} \right\rangle$$

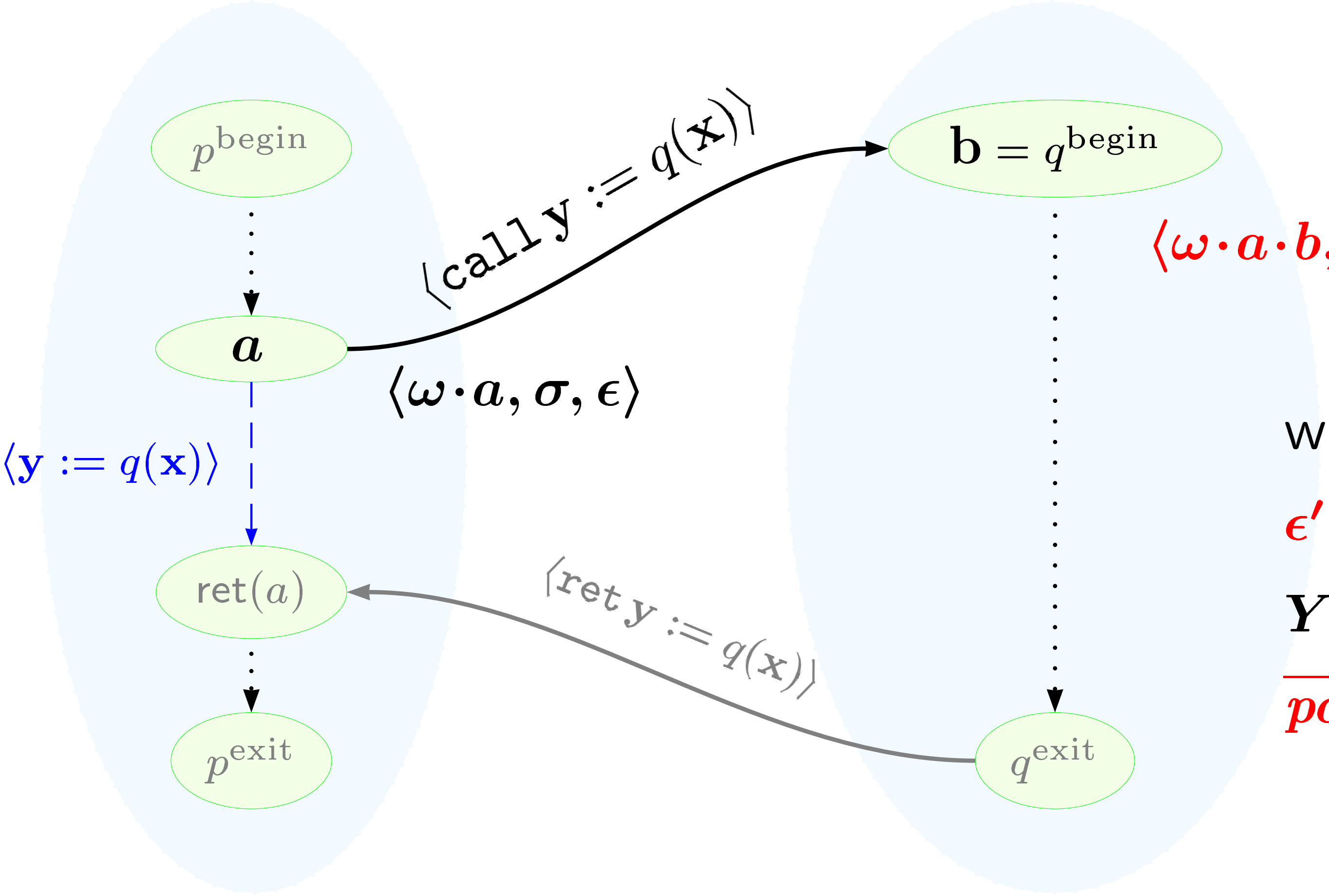
$$\left\langle \overline{\boldsymbol{A_1}}, \widetilde{\boldsymbol{A_1}} \right\rangle \sqsubseteq \left\langle \overline{\boldsymbol{A_2}}, \widetilde{\boldsymbol{A_2}} \right\rangle \text{ iff } \overline{\boldsymbol{A_1}} \subseteq \overline{\boldsymbol{A_2}} \text{ and } \widetilde{\boldsymbol{A_1}} \subseteq \widetilde{\boldsymbol{A_2}}$$

Abstract Forward Collecting Semantics – `call`



$$I^*(a,b) = \langle \texttt{call}\, \mathbf{y} := q(\mathbf{x})\rangle$$

Abstract Forward Collecting Semantics – `call`



where:

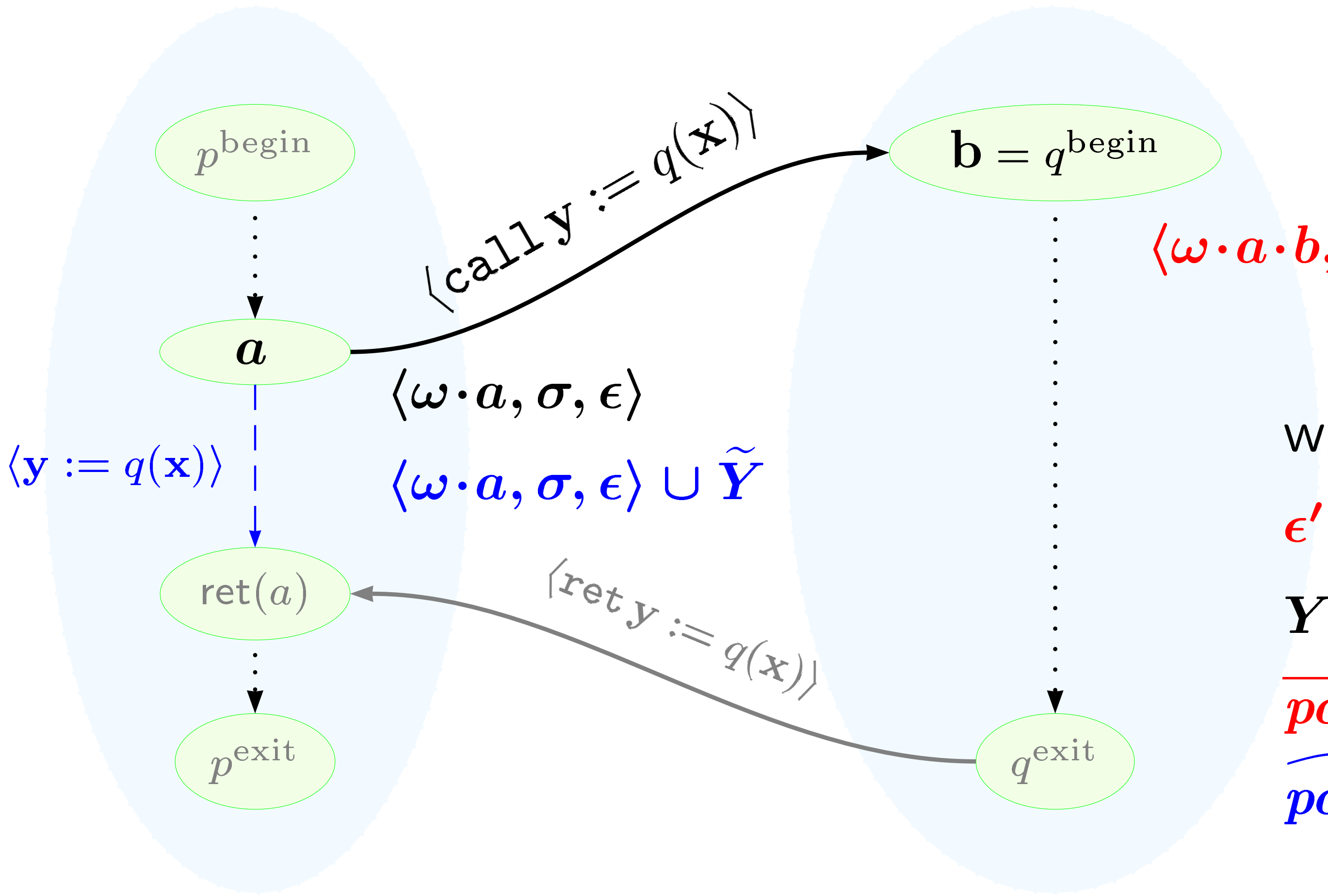
$\boldsymbol{\epsilon' = [\mathbf{fp}_q \mapsto (\sigma \oplus \epsilon)(\mathbf{x})]}$

$\boldsymbol{Y = \langle \overline{Y}, \widetilde{Y} \rangle}$

$\overline{\boldsymbol{post}^{\mathbf{a}}}\boldsymbol{(Y)}$

$I^*(a, b) = \langle \texttt{call}\, \mathbf{y} := q(\mathbf{x}) \rangle$

Abstract Forward Collecting Semantics – call



where:

$\epsilon' = [\mathrm{fp}_q \mapsto (\sigma \oplus \epsilon)(\mathbf{x})]$

$Y = \langle \overline{Y}, \widetilde{Y} \rangle$

$\overline{post^{\mathbf{a}}}(Y)$

$\widetilde{post^{\mathbf{a}}}(Y)$

$I^*(a, b) = \langle \texttt{call}\, \mathbf{y} := q(\mathbf{x})\rangle$

Abstract Forward Collecting Semantics – `call`

$$\overline{post^{\mathsf{a}}}(c \xrightarrow{\langle \mathtt{call}\, \mathbf{y}:=P_j(\mathbf{x})\rangle} c')(Y) =$$

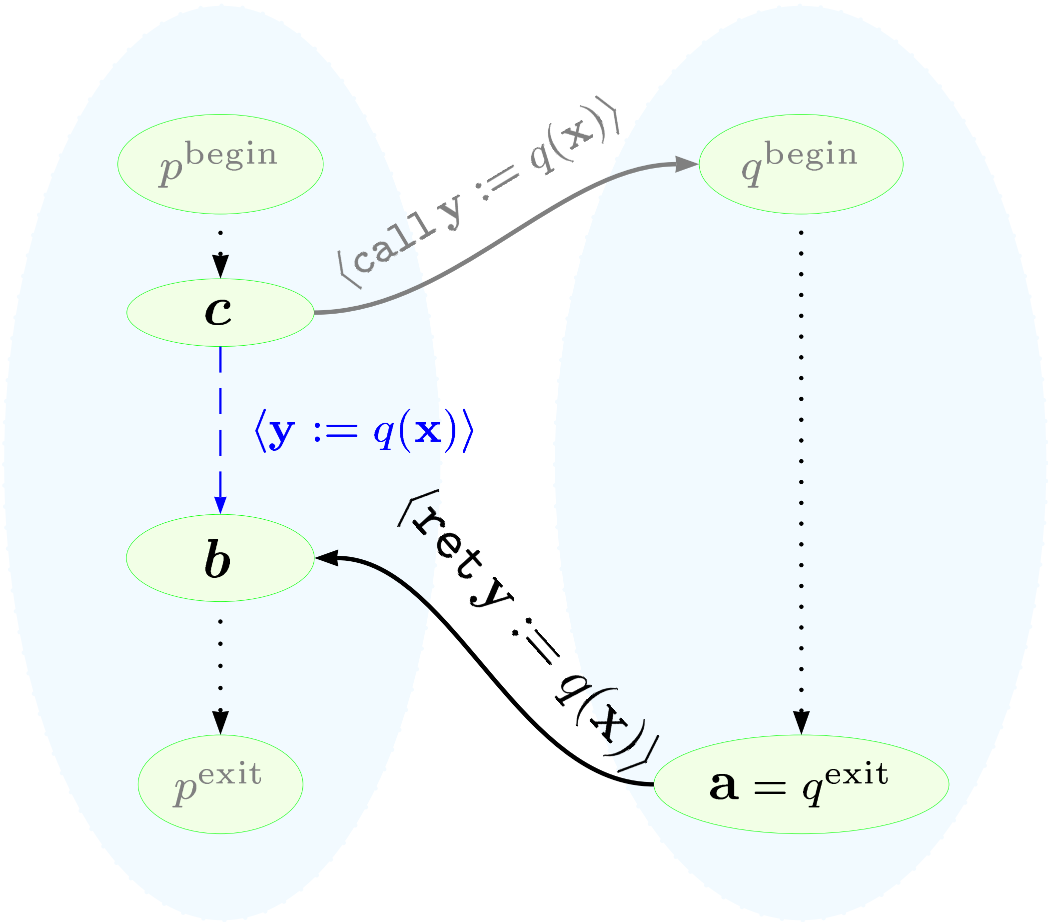
$$\left\{ \langle \omega \cdot c \cdot c', \sigma, \epsilon' \rangle \;\middle|\; \begin{array}{l} \langle \omega \cdot c, \sigma, \epsilon \rangle \in \overline{Y} \\ \epsilon' = [\mathbf{fp}_j^{(k)} \mapsto (\sigma \oplus \epsilon)(\mathbf{x}^{(k)})] \end{array} \right\}$$

$$\widetilde{post^{\mathsf{a}}}(c \xrightarrow{\langle \mathtt{call}\, \mathbf{y}:=P_j(\mathbf{x})\rangle} c')(Y) =$$

$$\widetilde{Y} \;\cup\; \left\{ \langle \omega \cdot c, \sigma, \epsilon \rangle \in \overline{Y} \right\}$$

$$post^{\mathsf{a}}(Y) = \left\langle \widetilde{post^{\mathsf{a}}}(Y), \overline{post^{\mathsf{a}}}(Y) \right\rangle$$

Abstract Forward Collecting Semantics – `ret`



$$I^*(a, b) = \langle \mathtt{ret}\, \mathbf{y} := q(\mathbf{x}) \rangle$$

Abstract Forward Collecting Semantics – `ret`

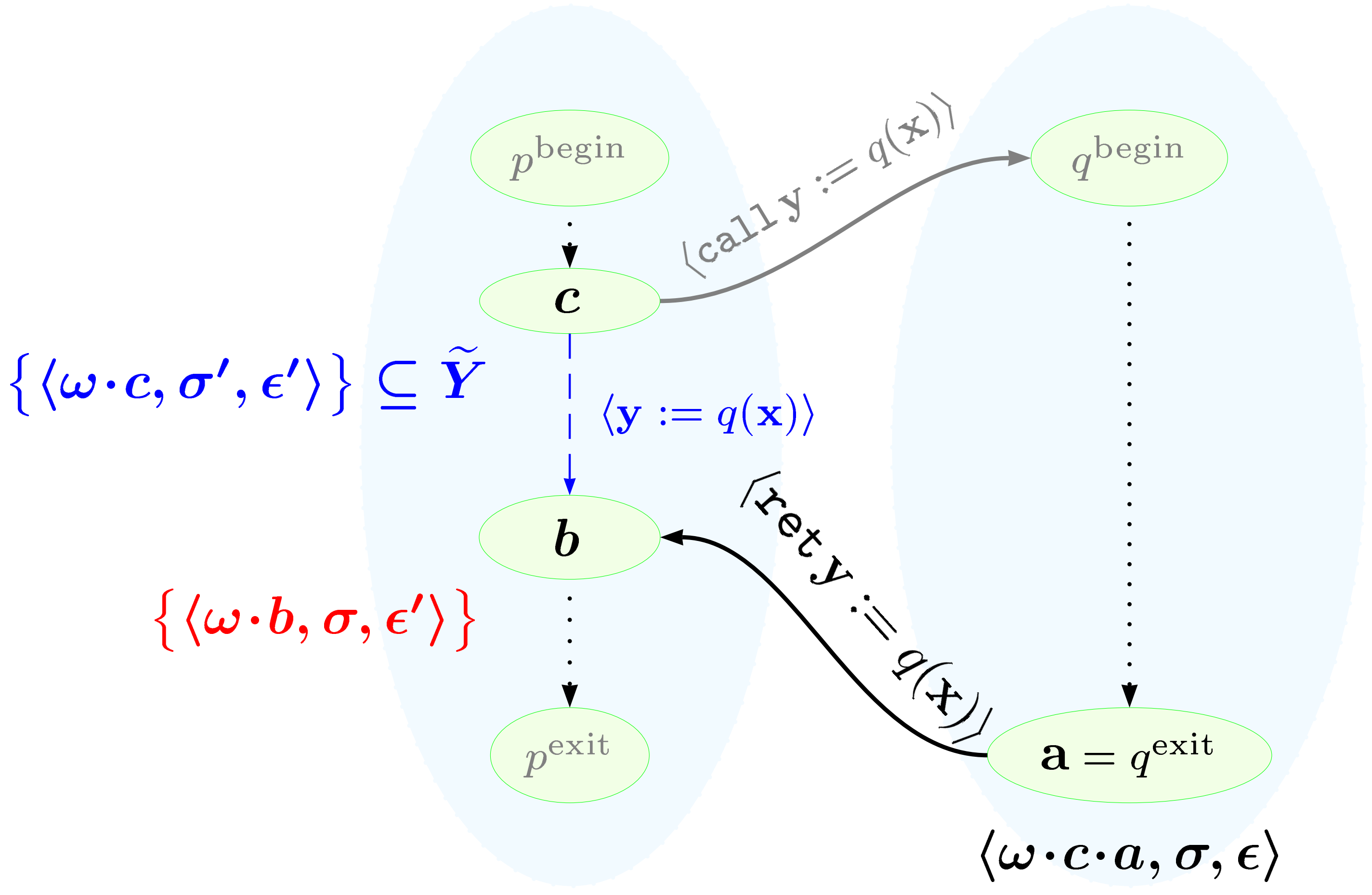
$\boxed{\mathbf{y} = \emptyset}$

where

$\boldsymbol{Y} = \langle \overline{\boldsymbol{Y}}, \widetilde{\boldsymbol{Y}} \rangle$

$\overline{\boldsymbol{post}}^{\mathbf{a}}(\boldsymbol{Y})$

$\boldsymbol{\epsilon}(\mathbf{fp}_q) = (\boldsymbol{\sigma}' \oplus \boldsymbol{\epsilon}')(\mathbf{x})$



$$I^*(a,b) = \langle \texttt{ret}\ \mathbf{y} := q(\mathbf{x})\rangle$$

Abstract Forward Collecting Semantics – ret

where

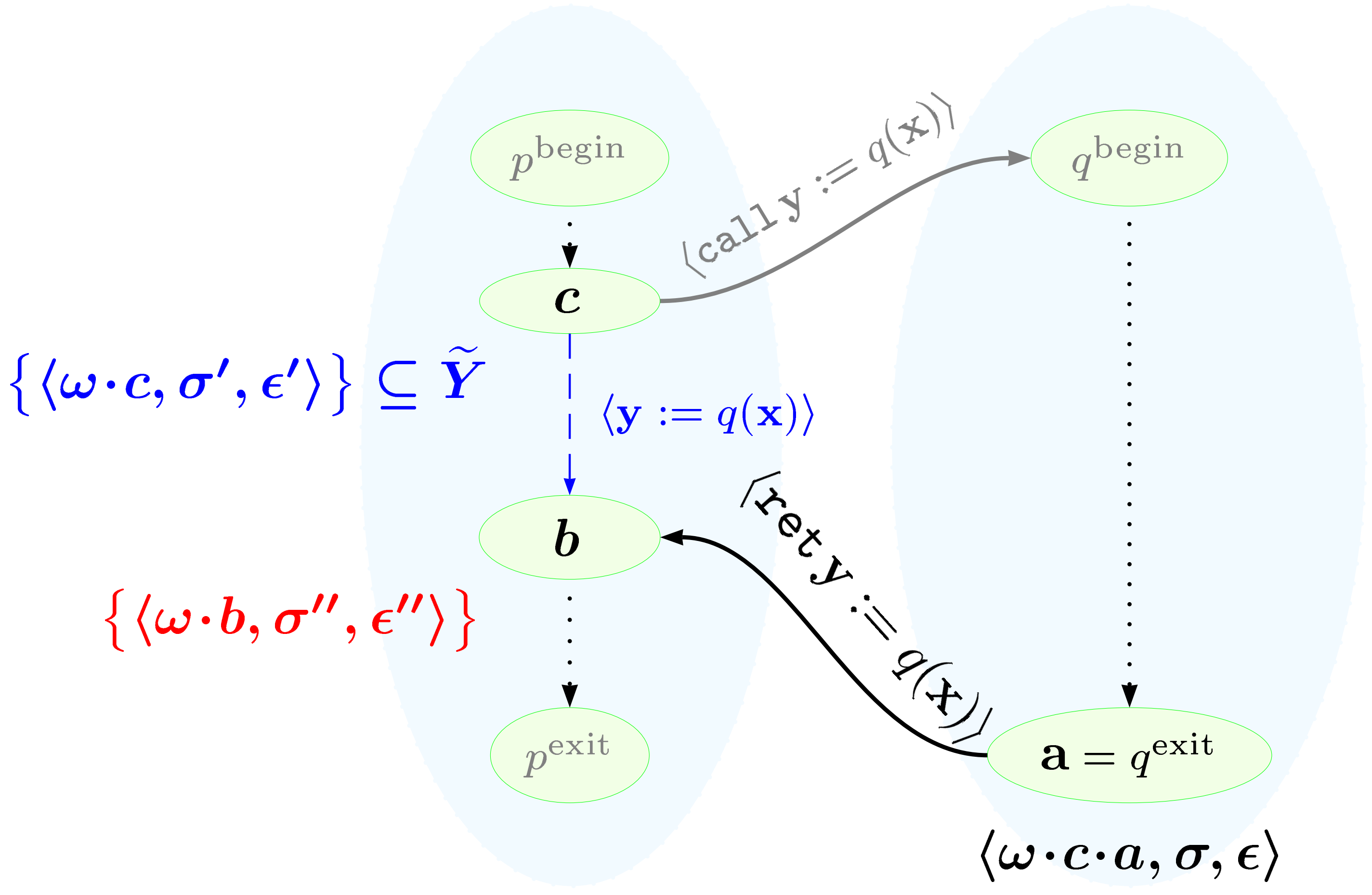
$\boldsymbol{Y} = \langle \overline{\boldsymbol{Y}}, \widetilde{\boldsymbol{Y}} \rangle$

$\overline{\boldsymbol{post}^{\mathbf{a}}}(\boldsymbol{Y})$

$\boldsymbol{\epsilon}(\mathbf{fp}_q) = (\boldsymbol{\sigma}' \oplus \boldsymbol{\epsilon}')(\mathbf{x})$

$\boldsymbol{\sigma}'' = \boldsymbol{\sigma}\left[\mathrm{y} \mapsto \boldsymbol{\epsilon}(\mathbf{fr}_q)\right]$

$\boldsymbol{\epsilon}'' = \boldsymbol{\epsilon}'\left[\mathrm{y} \mapsto \boldsymbol{\epsilon}(\mathbf{fr}_q)\right]$



$$I^*(a,b) = \langle \mathtt{ret}\, \mathbf{y} := q(\mathbf{x}) \rangle$$

Abstract Forward Collecting Semantics – `ret`

$$\overline{post^{\mathsf{a}}}(c \xrightarrow{\langle \texttt{ret}\,\mathbf{y}{:=}P_j(\mathbf{x})\rangle} c')(Y) =$$

$$\left\{ \langle \omega \cdot c', \sigma, \epsilon \rangle \;\middle|\; \begin{array}{l} \langle \omega \cdot \mathsf{call}(c') \cdot c, \overline{\sigma}, \overline{\epsilon} \rangle \in \overline{Y} \\ \langle \omega \cdot \mathsf{call}(c'), \widetilde{\sigma}, \widetilde{\epsilon} \rangle \in \widetilde{Y} \\ (\widetilde{\sigma} \oplus \widetilde{\epsilon})(\mathbf{x}^{(k)}) = \overline{\epsilon}(\mathbf{fp}_j^{(k)}) \\ \sigma = \overline{\sigma}[\mathbf{y}^{(k)} \mapsto \overline{\epsilon}(\mathbf{fr}_j^{(k)}) \mid \mathbf{y}^{(k)} \in GVar] \\ \epsilon = \widetilde{\epsilon}[\mathbf{y}^{(k)} \mapsto \overline{\epsilon}(\mathbf{fr}_j^{(k)}) \mid \mathbf{y}^{(k)} \notin GVar] \end{array} \right\}$$

$$\widetilde{post^{\mathsf{a}}}(c \xrightarrow{\langle \texttt{ret}\,\mathbf{y}{:=}P_j(\mathbf{x})\rangle} c')(Y) =$$

$$\left\{ \langle \omega', \sigma', \epsilon' \rangle \in \widetilde{Y} \mid \exists \omega \in \mathit{Ctrl}^{+} : \langle \omega' \cdot \omega \cdot c, \sigma, \epsilon \rangle \in \overline{Y} \right\}$$

$$post^{\mathsf{a}}(Y) = \left\langle \widetilde{post^{\mathsf{a}}}(Y), \overline{post^{\mathsf{a}}}(Y) \right\rangle$$

Abstraction Function: α

on singletons:

$$\alpha(\{x\}) \stackrel{\text{def}}{=} \langle \overline{\alpha}(\{x\}),\ \widetilde{\alpha}(\{x\}) \rangle$$

$$\overline{\alpha}\big(\{\langle \sigma_n,\ \langle c_0, \epsilon_0 \rangle \cdot \ldots \cdot \langle c_n, \epsilon_n \rangle \rangle\}\big) \stackrel{\text{def}}{=} \{\langle c_0 \cdot \ldots \cdot c_n,\ \sigma_n,\ \epsilon_n \rangle\},$$

$$\widetilde{\alpha}\big(\{\langle \sigma_n,\ \langle c_0, \epsilon_0 \rangle \cdot \ldots \cdot \langle c_n, \epsilon_n \rangle \rangle\}\big) \stackrel{\text{def}}{=}$$

$$\left\{ \langle c_0 \cdot \ldots \cdot c_i,\ \sigma_i,\ \epsilon_i \rangle \;\middle|\; \begin{array}{l} 0 \le i < n \text{ and } \forall i < n : \\ \langle c_i, \sigma_i, \epsilon_i \rangle \text{ valid for } \langle c_{i+1}, \sigma_{i+1}, \epsilon_{i+1} \rangle \end{array} \right\}$$

on sets of states:

$$\alpha(X) \stackrel{\text{def}}{=} \bigsqcup_{x \in X} \alpha(\{x\})$$

Property:

$\langle c_0 \cdot \ldots \cdot c_n,\ \sigma,\ \epsilon \rangle \in \widetilde{Y} \quad \Rightarrow \quad \forall i$ with $0 \le i \le n :\ \ c_i$ is a call-site

Concretization Function: γ

$$\left\langle \boldsymbol{\sigma}_n,\ \langle c_0, \epsilon_0\rangle \cdot \ldots \cdot \langle c_n, \epsilon_n\rangle \right\rangle \in \gamma(Y)$$

$$\Leftrightarrow$$

$$\begin{cases} \langle c_0 \cdot \ldots \cdot c_n,\ \boldsymbol{\sigma}_n,\ \epsilon_n\rangle \in \overline{Y} \\ \exists \boldsymbol{\sigma}_0, \ldots, \boldsymbol{\sigma}_{n-1} : \forall i < n : \begin{cases} \langle c_0 \cdot \ldots \cdot c_i,\ \boldsymbol{\sigma}_i,\ \epsilon_i\rangle \in \widetilde{Y} \text{ and} \\ \langle c_i, \boldsymbol{\sigma}_i, \epsilon_i\rangle \text{ valid for } \langle c_{i+1}, \boldsymbol{\sigma}_{i+1}, \epsilon_{i+1}\rangle \end{cases} \end{cases}$$

Correctness of the Abstraction

Theorem 1 $post^{\mathsf{a}} \sqsupseteq \alpha \circ post \circ \gamma$

Corollary 2 $\gamma\big(reach^{\mathsf{a}}(Y_0)\big) \sqsupseteq reach\big(\gamma(Y_0)\big)$

Exact Analysis if $GVar = \emptyset$

Theorem 3 $GVar = \emptyset \quad \Longrightarrow \quad \gamma \circ reach^{\mathsf{a}} = reach \circ \gamma$

$reach : \wp(State) \to \wp(State)$ reachable states

$reach^{\mathsf{a}}: \quad A \to A$ reachable abstract states

Information Lost by Our Abstraction

"relationships between different local environments in a call stack"

$$(\gamma \circ \alpha)\left(\left\{\begin{array}{l} \langle \sigma,\ \langle c_0, \epsilon_0 \rangle \cdot \langle c_1, \epsilon_1 \rangle \rangle, \\ \langle \sigma,\ \langle c_0, \epsilon'_0 \rangle \cdot \langle c_1, \epsilon'_1 \rangle \rangle \end{array}\right\}\right) = \left\{\begin{array}{l} \langle \sigma,\ \langle c_0, \epsilon_0 \rangle \cdot \langle c_1, \epsilon_1 \rangle \rangle, \\ \langle \sigma,\ \langle c_0, \epsilon'_0 \rangle \cdot \langle c_1, \epsilon_1 \rangle \rangle, \\ \langle \sigma,\ \langle c_0, \epsilon_0 \rangle \cdot \langle c_1, \epsilon'_1 \rangle \rangle, \\ \langle \sigma,\ \langle c_0, \epsilon'_0 \rangle \cdot \langle c_1, \epsilon'_1 \rangle \rangle \end{array}\right\}$$

But: the states have to be *valid*:

$\epsilon_0(\mathbf{x}) = \epsilon'_1(\mathbf{fp})$ and $\epsilon'_0(\mathbf{x}) = \epsilon_1(\mathbf{fp})$

Abstract Interpretation: Abstract Domains – 1

$$A \;\simeq\; Ctrl \;\to\; \left(\; \boxed{\wp(Ctrl^* \times GEnv \times LEnv)} \;\right)^2$$

$Ctrl$ is a finite set,
but how to abstract ?

If $GEnv \simeq \mathbb{B}^{m_g} \times \mathbb{N}^{n_g}$ and $LEnv \simeq \mathbb{B}^{m_l} \times \mathbb{N}^{n_l}$ then

$$\wp(Ctrl^* \times GEnv \times LEnv) \;\simeq\; \wp(Ctrl^* \times \mathbb{B}^{m_g+m_l} \times \mathbb{N}^{n_g+n_l})$$

Use the following abstraction

$$\boxed{\mathcal{A}^{(1)}(Ctrl^*) \times \mathcal{A}^{(m_g+m_l)}(\mathbb{B}) \times \mathcal{A}^{(n_g+n_l)}(\mathbb{N})}$$

where $\mathcal{A}^{(k)}(S)$ is an abstract domain for $\wp(S^k)$

represent *relations* between different domains by *partitioning*

Abstract Interpretation: Abstract Domains – 2

Booleans: no need for abstraction

Numerics:

- Int$(\mathbb{N}^k)$: lattice of intervals
- Oct$(\mathbb{N}^k)$: lattice of octagons
- Pol$(\mathbb{N}^k)$: lattice of convex polyhedrons

Stacks:

- $\wp(Ctrl^n)$: keep n top elements, forget the rest
- Reg$(Ctrl)$: lattice of regular languages over $Ctrl$ (+ widening)
- "CallGraph-FSA": finite state automatons with states $= Ctrl$

relational domains: select "matching" calling contexts on returns

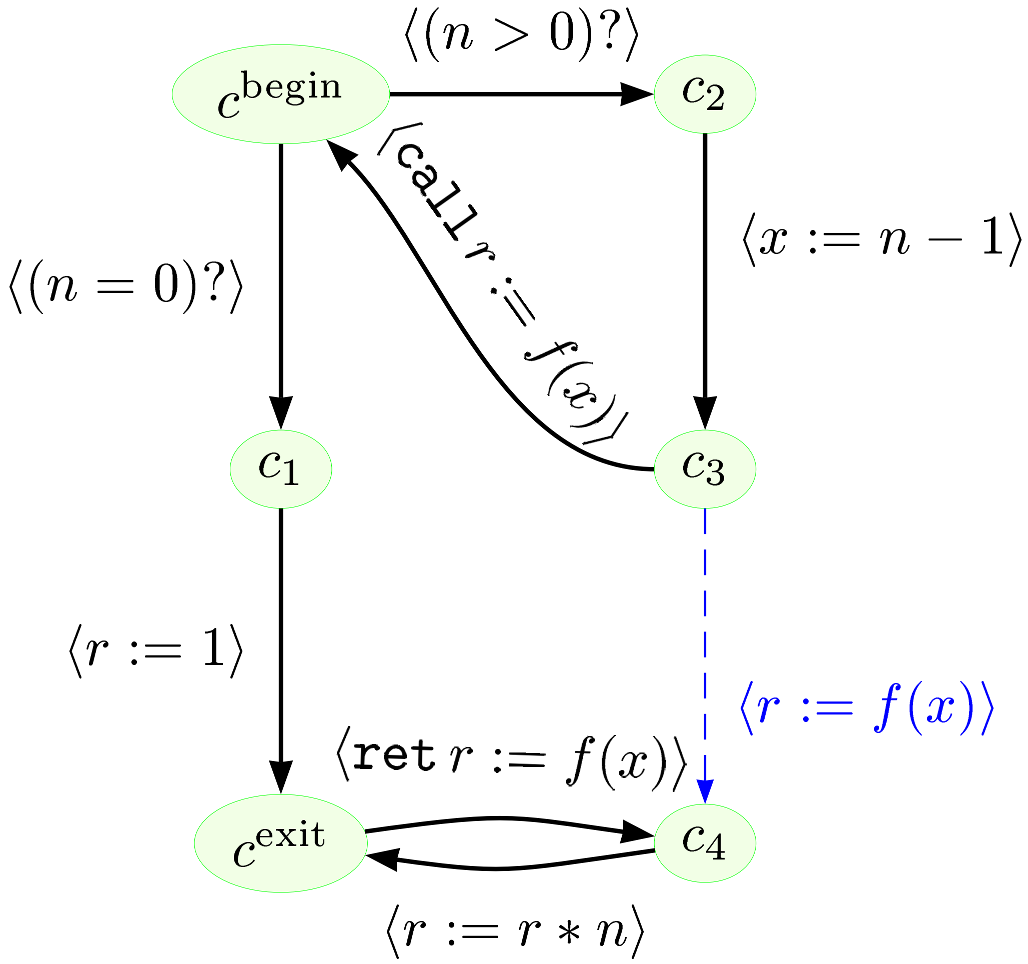
Conclusion

- abstract semantics of programs with
  - recursive procedures
  - local and global variables
- no loss of precision if no global variables
- known abstract interpretation techniques for effective analysis

Perspectives

- Implementation: Under Work (Integration & Test)
- Extension to the verification of security properties
- Heap for dynamic memory allocation

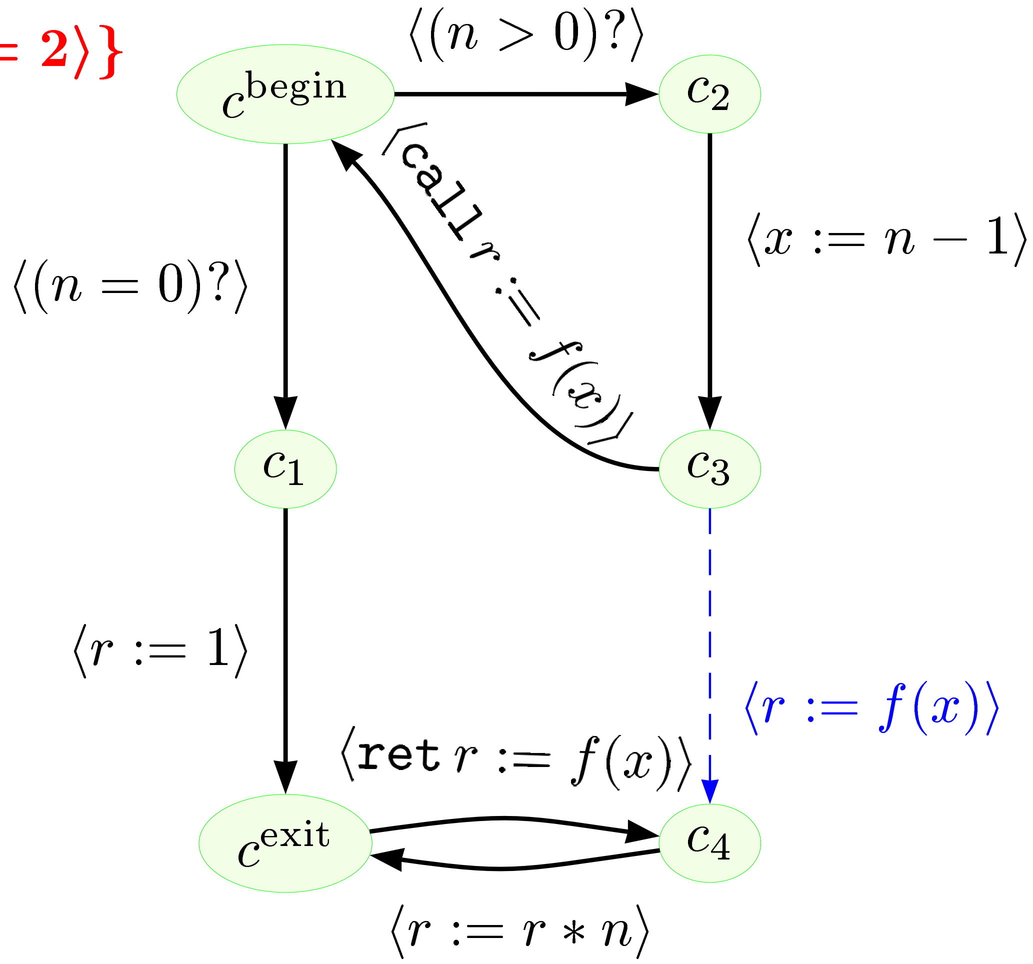
Analysis: Example



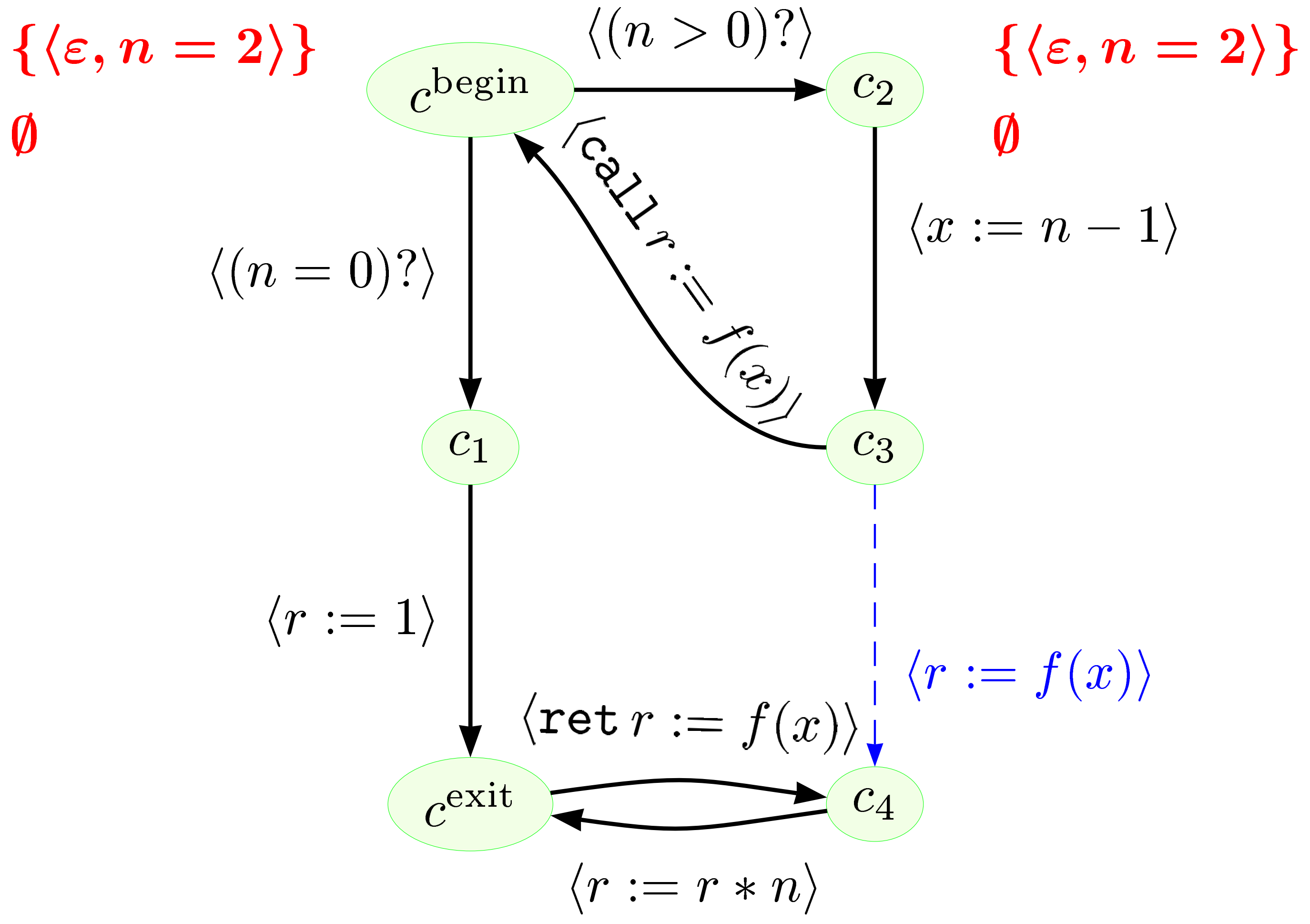
Analysis: Example

$\{\langle \varepsilon, n = 2 \rangle\}$

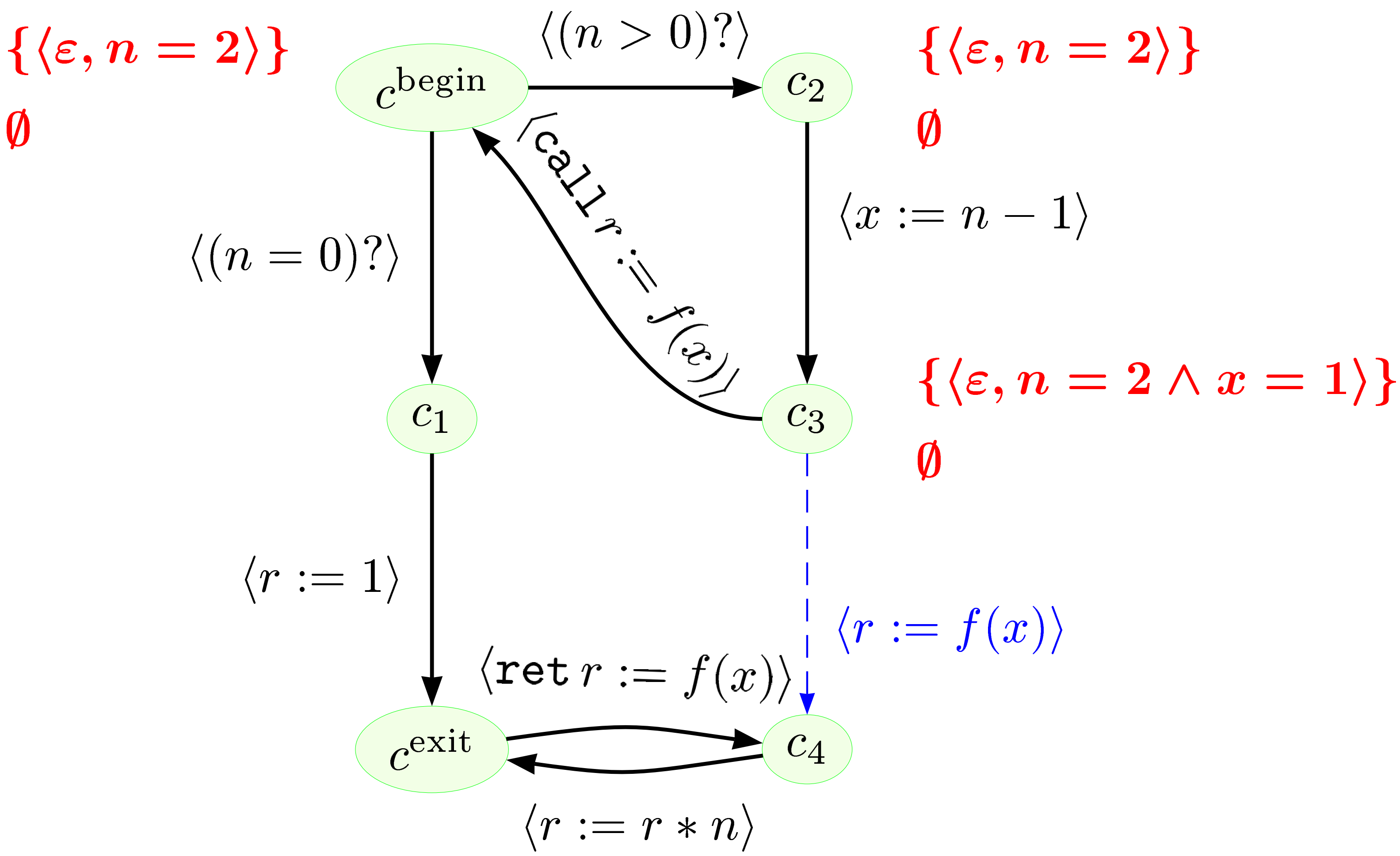
$\emptyset$



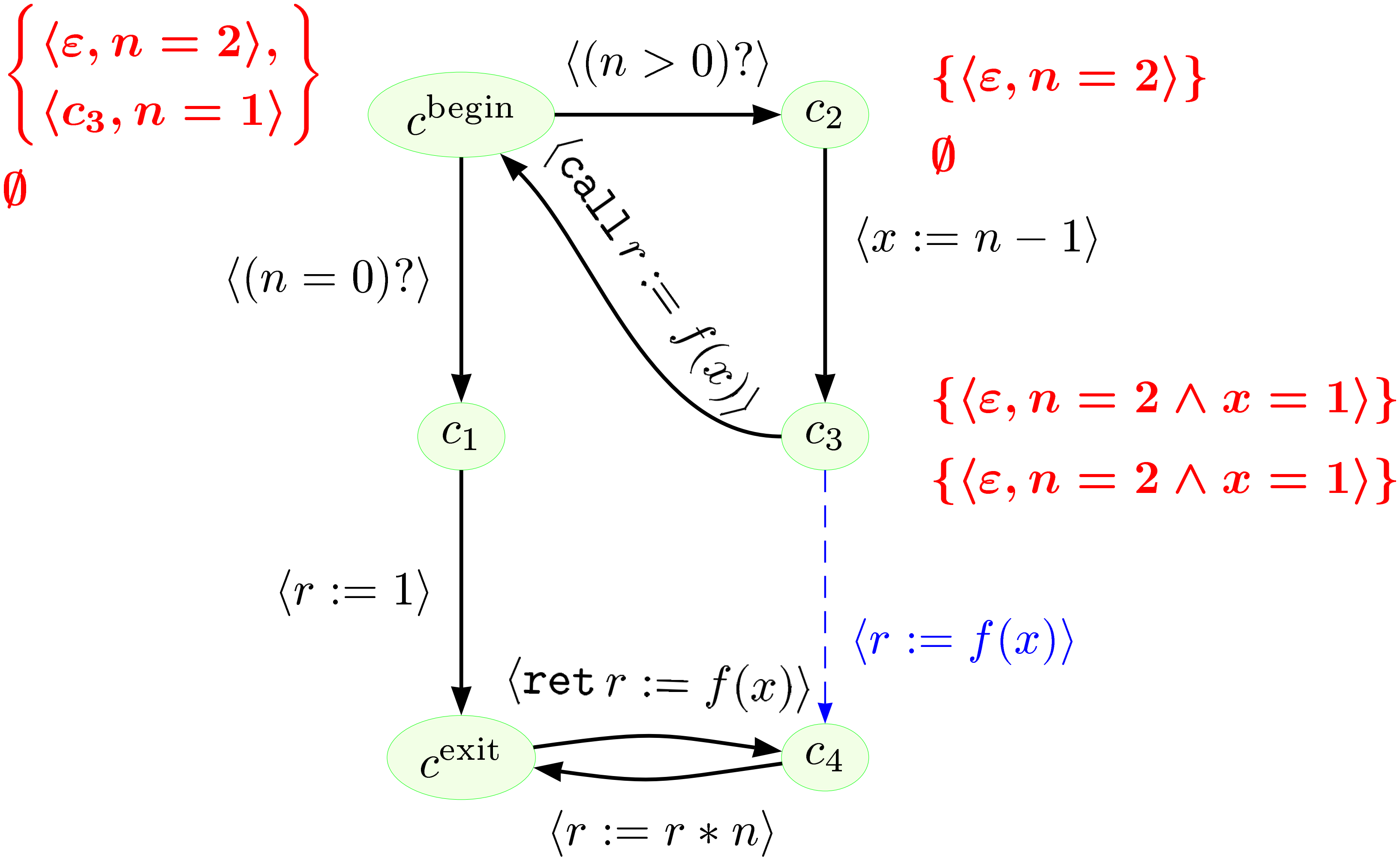
Analysis: Example



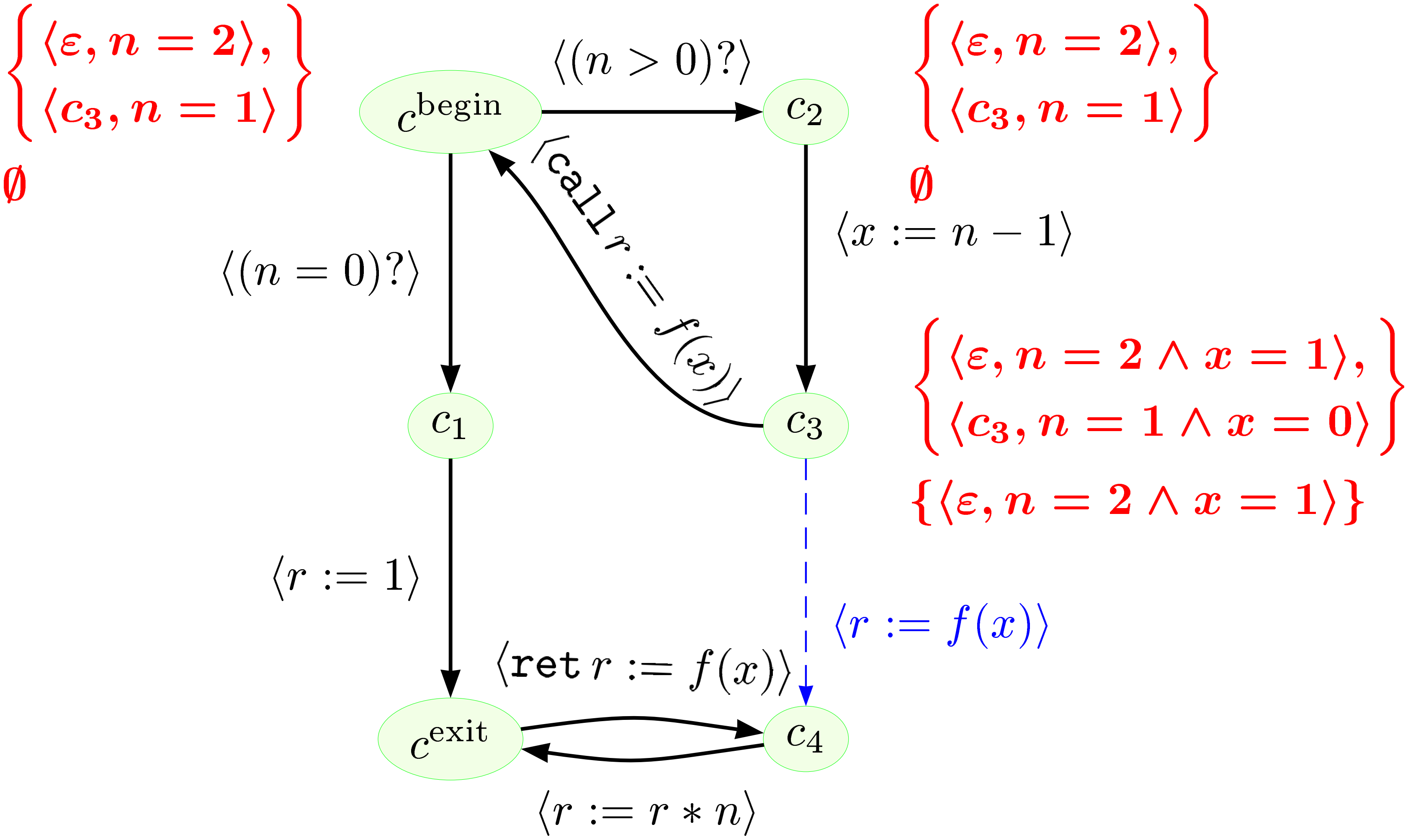
Analysis: Example



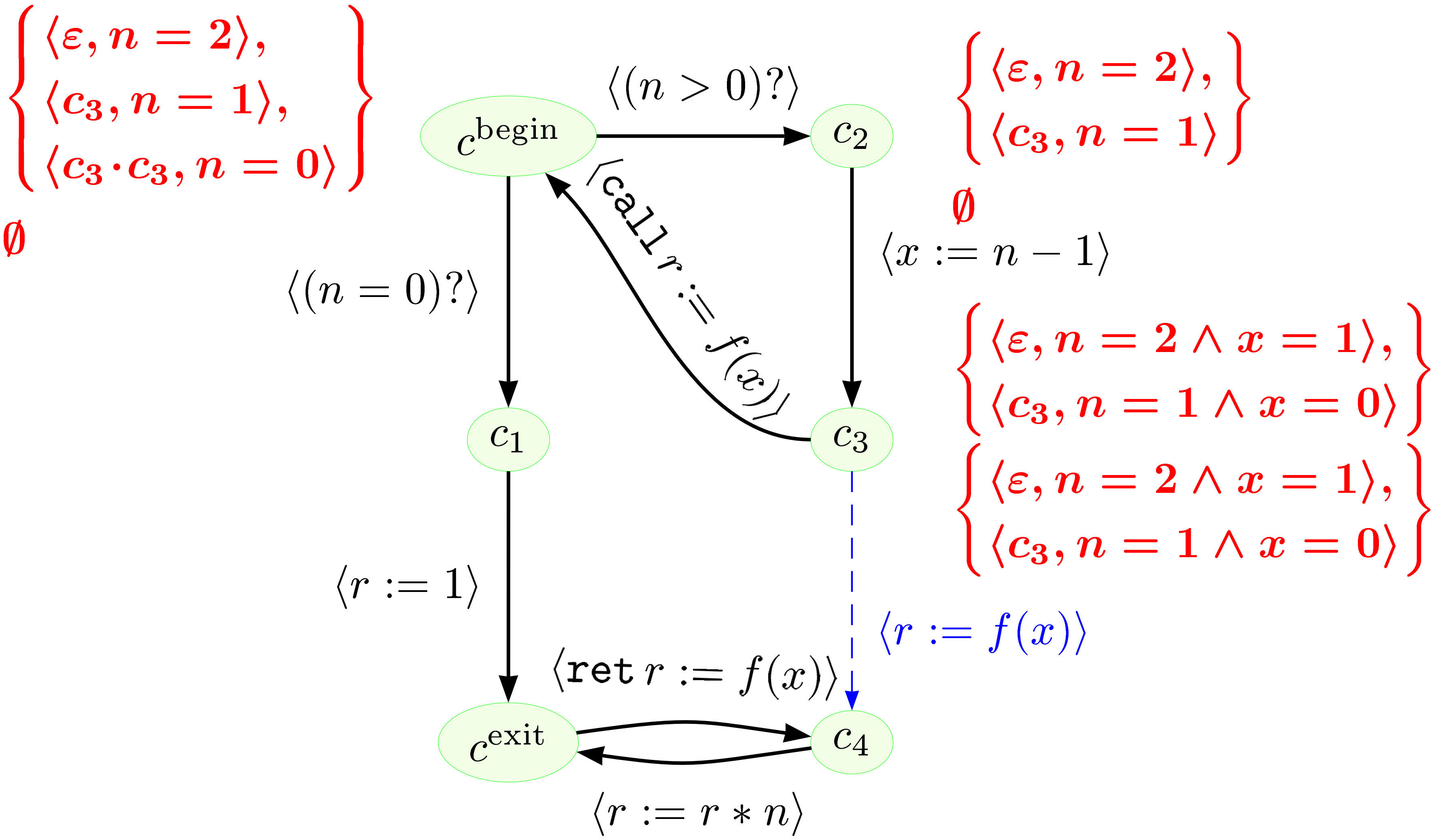
Analysis: Example



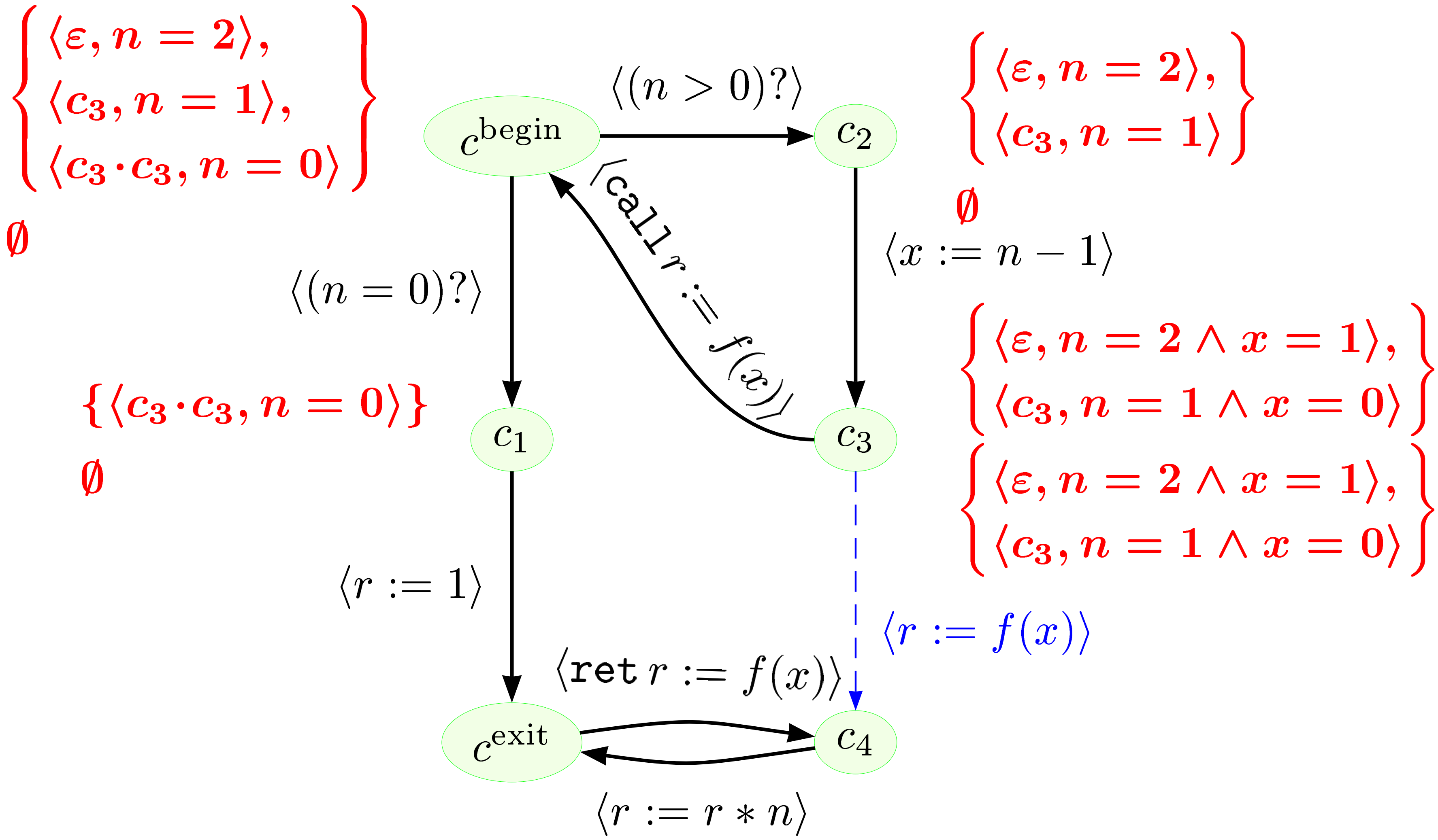
Analysis: Example



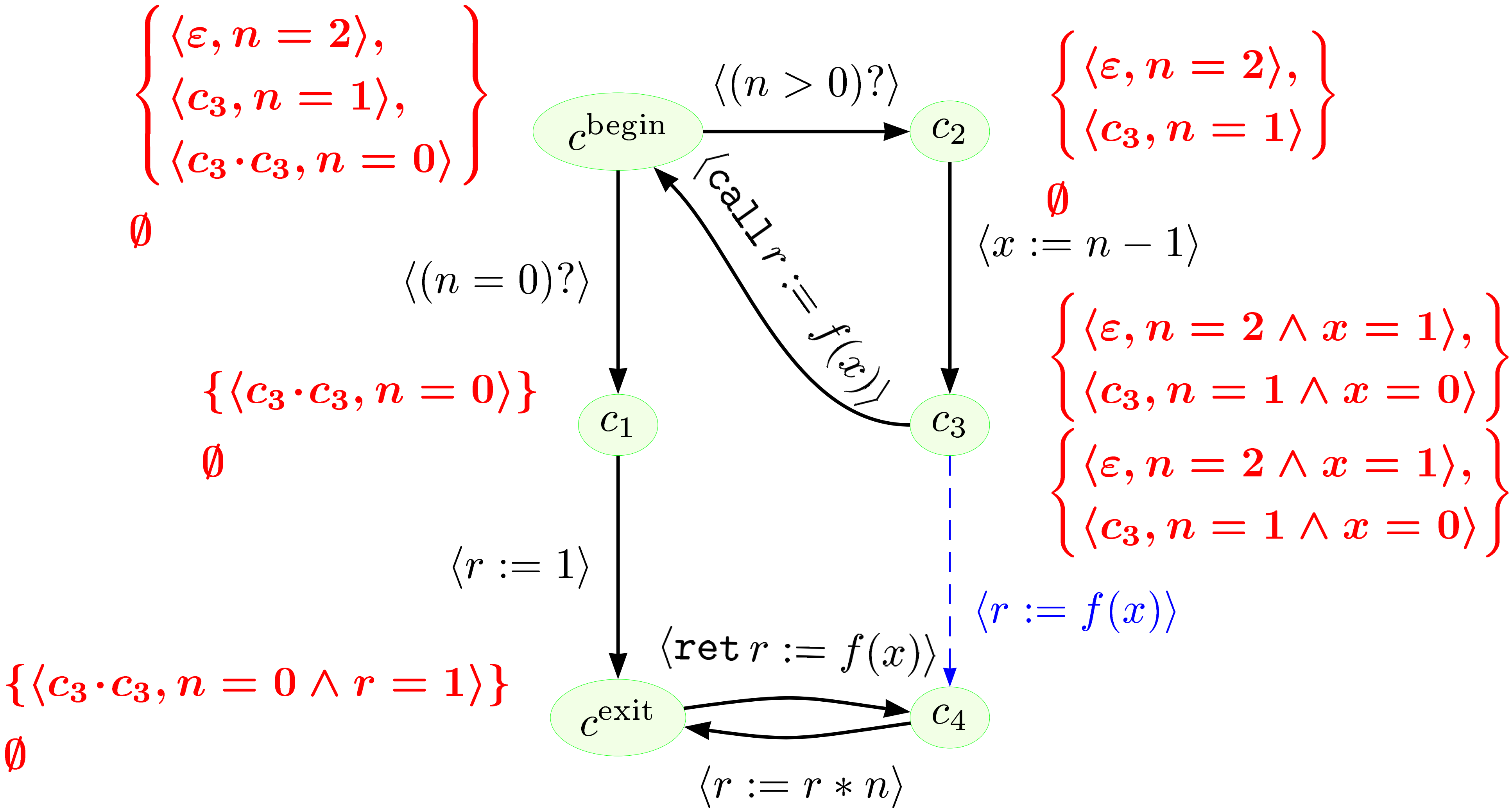
Analysis: Example



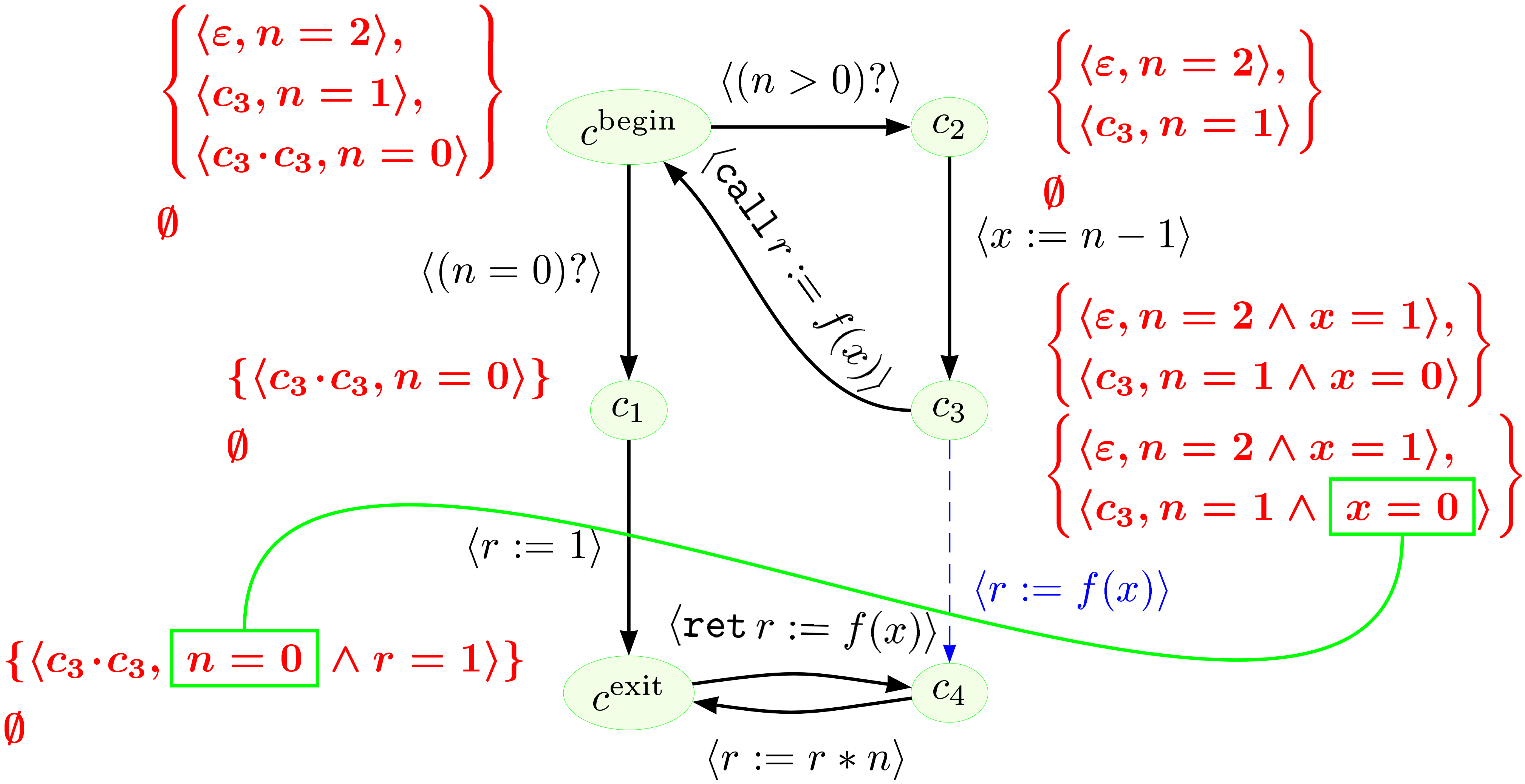
Analysis: Example



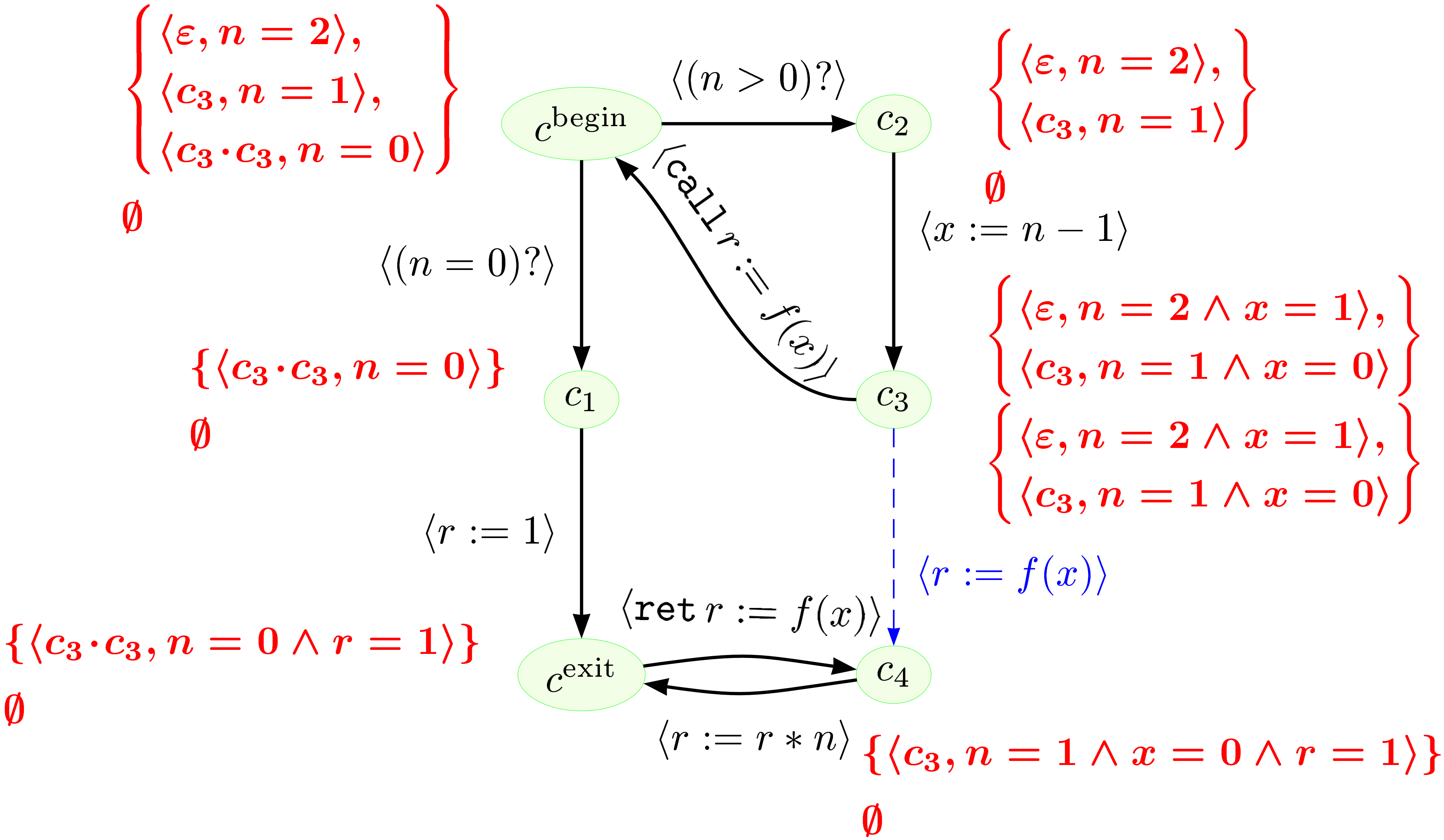
Analysis: Example



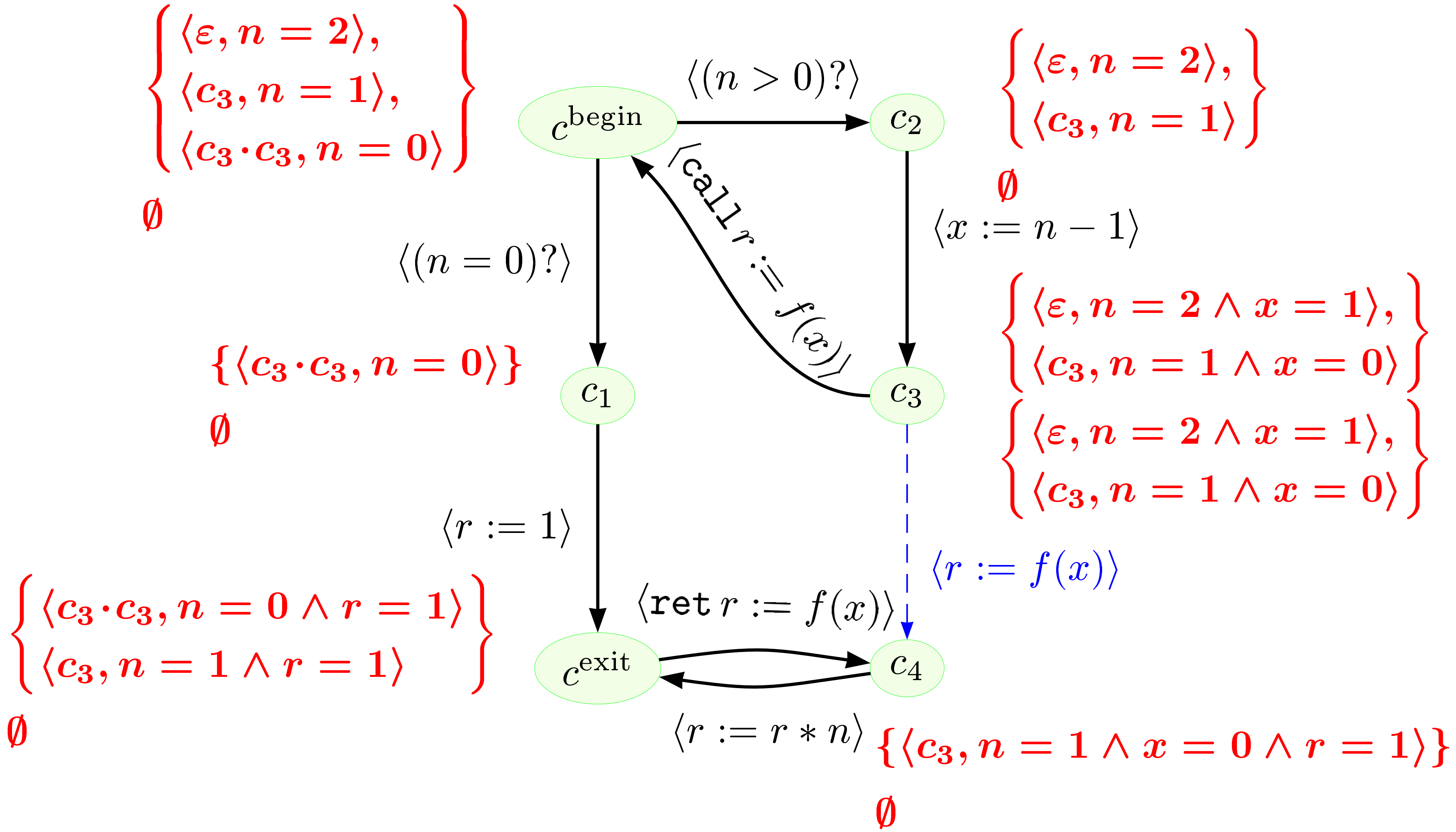
Analysis: Example



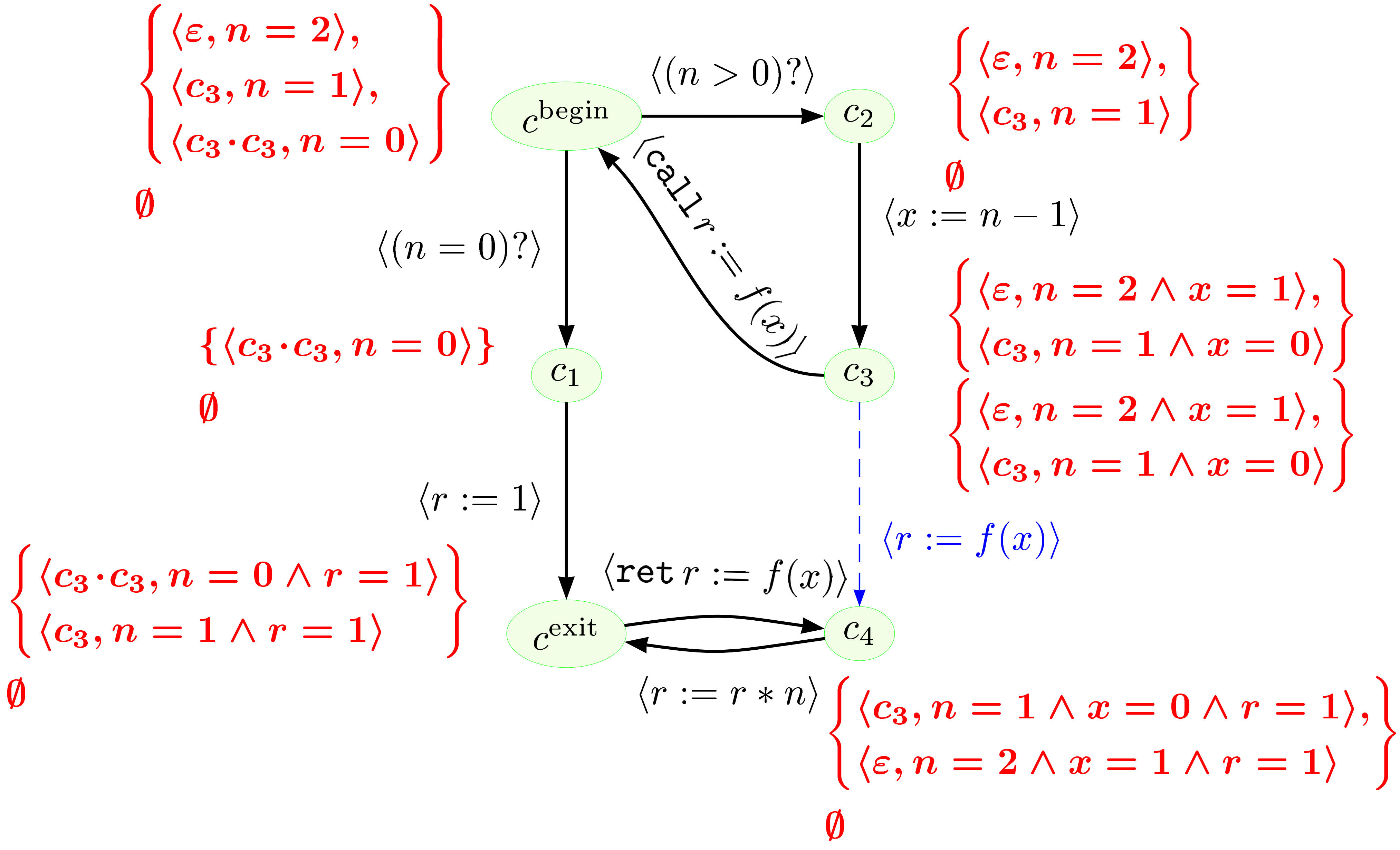
Analysis: Example



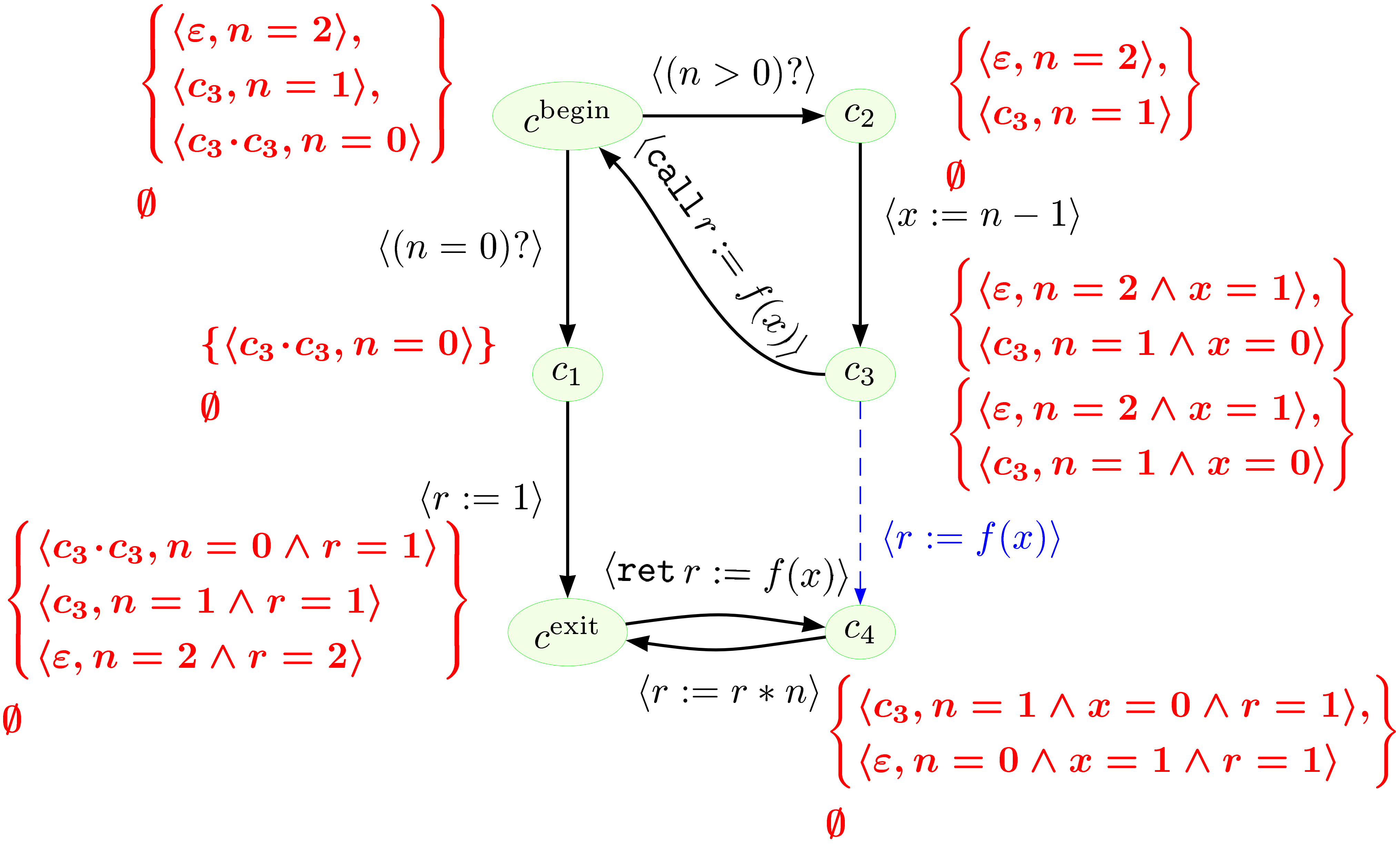
Analysis: Example



Analysis: Example



Analysis: Example



Analysis: Example