Análisis estático para el MIDP

Luis Sierra

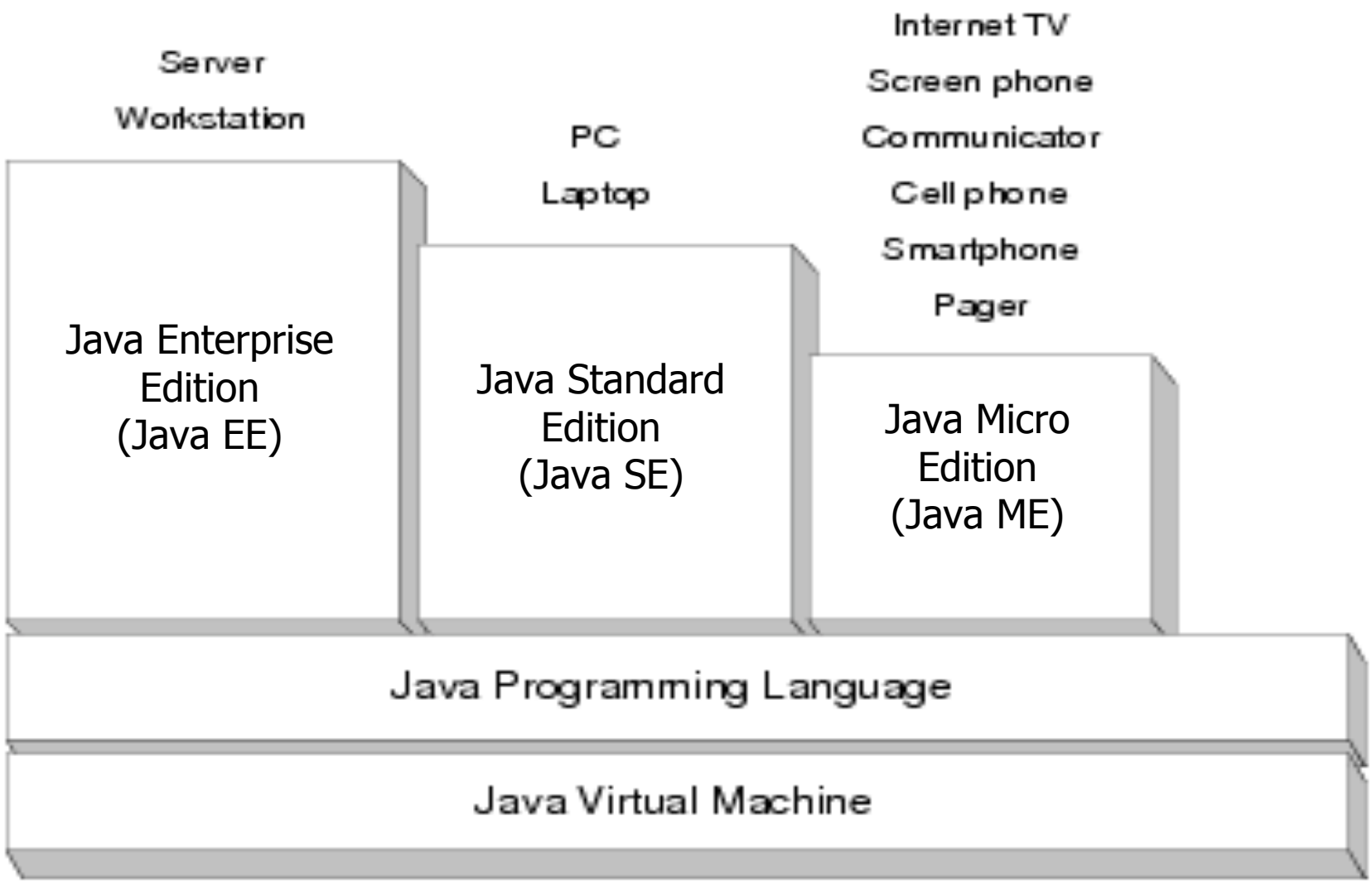
Octubre del 2007

Rosario, Argentina

Plan

- Problema: seguridad en MIDP
- Formalización: propuesta de Besson, Dufay, y Jensen
- Herramienta: análisis estático
- Propuestas

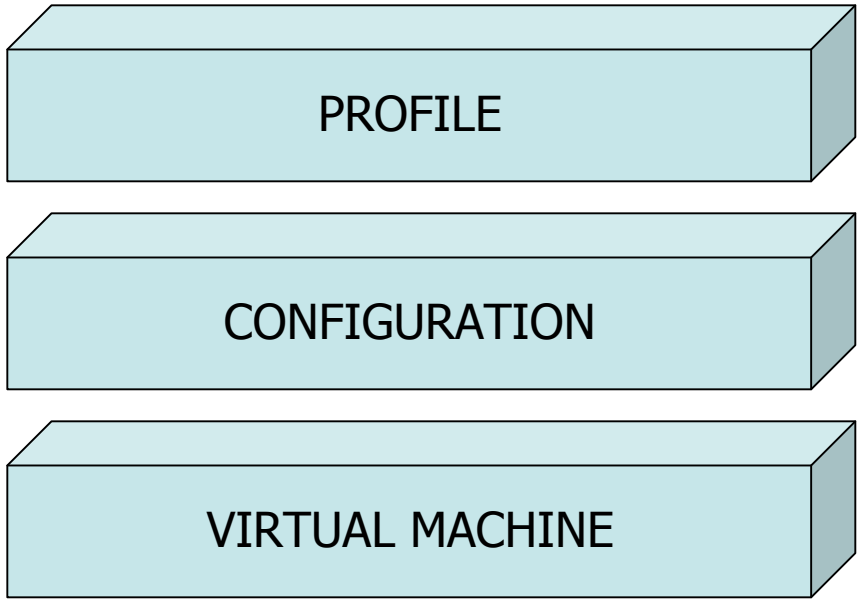
Java's


Server
Workstation
PC
Laptop
Internet TV
Screen phone
Communicator
Cell phone
Smartphone
Pager
Java Enterprise Edition (Java EE)
Java Standard Edition (Java SE)
Java Micro Edition (Java ME)
Java Programming Language
Java Virtual Machine

JME

- Diseñado con vistas a uso en dispositivos móviles y embebidos
  - restricciones de memoria
  - conectividad
- Flexibilidad
  - gran variedad de dispositivos diferentes
  - Tecnología en continuo cambio
  - Requerimientos de usuarios aún no estables
    - (o quizá inherentemente inestables)

Arquitectura: tres capas



El perfil es un conjunto de APIs (Application Programming Interfaces) para una configuración particular. Un dispositivo puede tener varios perfiles. MIDP es el perfil para CLDC.

Restricciones Diferentes
CLDC (Connected Limited Device Conf.)
CDC (Connected Device Configuration)

Implementaciones estándar de Sun
KVM (Kilo Virtual Machine)
CLDC Hotspot
CVM (Connected Virtual Machine)

MIDP: Mobile Information Device Profile

- MIDP es una plataforma para el desarrollo y difusión de aplicaciones gráficas y de red (MIDlets).
- Un dispositivo con MIDP puede
  - navegar por una lista de MIDlets en un servidor web
  - elegir uno de ellos
  - bajar, instalar y ejecutar
- Un MIDlet puede ejecutarse online u offline, actualizarse o eliminarse

Clasificacion de las APIs

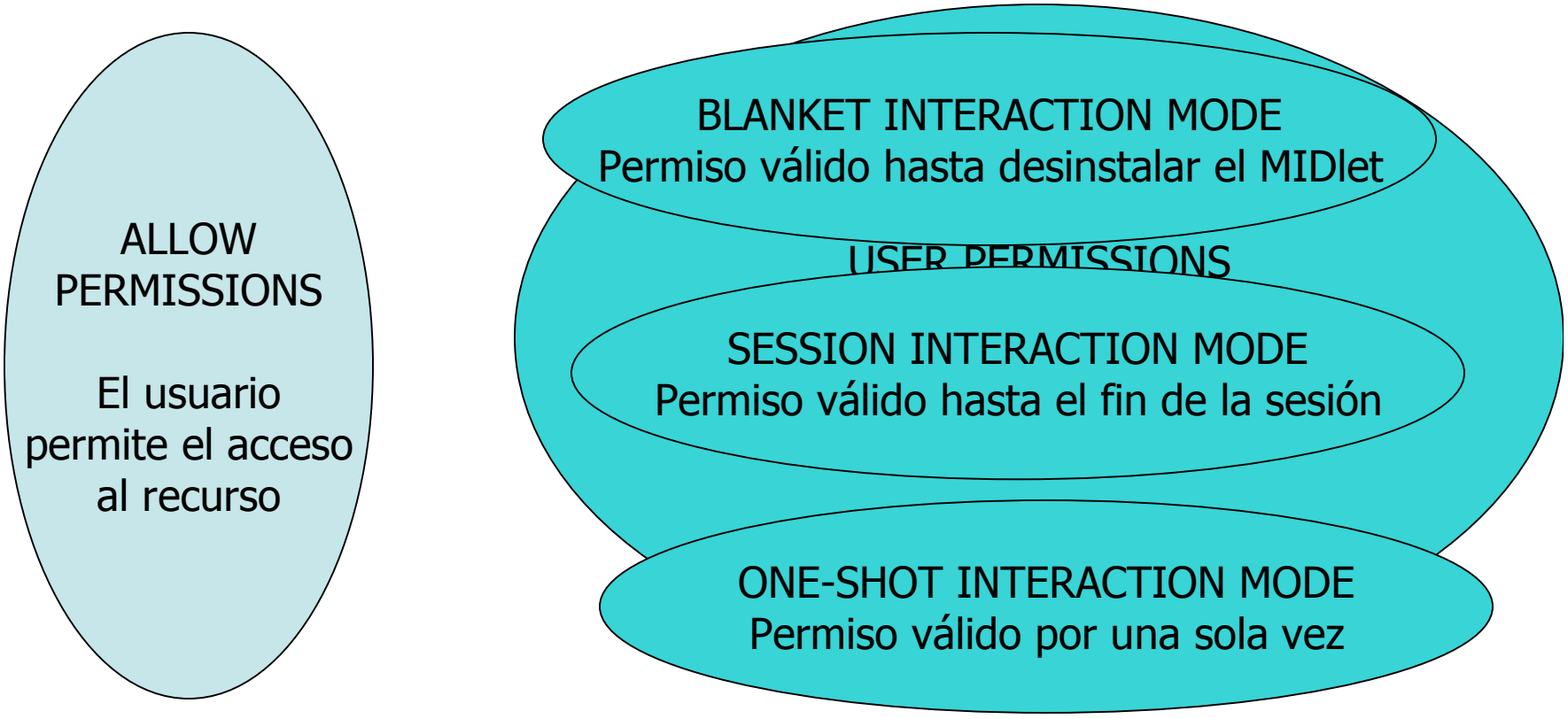
- de gestión de aplicaciones (AMS: Application Management System), responsables de la instalación, actualización y eliminación de MIDlets.
- de conectividad, responsables de establecer conexiones de red
- de interfase de usuario, que proveen componentes GUI (botones, cajas de texto) con sus manejadores de eventos
- de multimedia y juegos
- de almacenamiento local para gestionar información permanente
- seguridad end-to-end para proteger el dispositivo de ataques

Modelo de seguridad

- Basado en dominios de protección
- Un MIDlet se asocia a un dominio al cargarse
- El dominio define los permisos que puede adquirir el MIDlet

Dominios de protección

- Conjunto de permisos y modos de interacción



Propuesta de Besson, Dufay y Jensen

- Flexibilizar la definición del dominio de protección
  - Los permisos pueden pedirse en cualquier momento de la ejecución
- Incorporar multiplicidades a los permisos
  - allowed, blanket, session se dan al inicio con multiplicidad infinito
  - oneshot se da explícitamente justo antes de un consumo

Permisos y multiplicidades

$$Mul := \langle \mathbb{N} \cup \{\bot_{Mul}, \infty\}, \leq \rangle$$

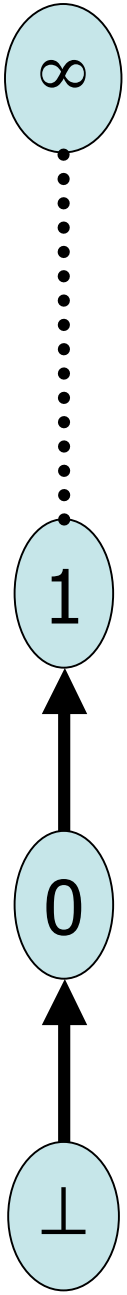
$$Perm_r := (\wp.Res_r \times \wp.Act_r) \cup \{\bot\}$$

Para cada tipo de recurso

Conjuntos de acciones y recursos

Permiso inválido

$$Perm := \prod_{r \in ResType} Perm_r \times Mul$$



Ejemplo

- El permiso p tal que
  - p.SMS = ((+1800*, {send}), 2)

Reticulado de permisos (1)

$$
\begin{aligned}
\rho_1 \leq \rho_2 \quad := \quad & (\forall r \in ResType : \rho_1.r \neq \bot : \\
& \rho_2.r \neq \bot \\
& \wedge\, \pi_1.\rho_1.r \subseteq \pi_1.\rho_2.r \\
& \wedge\, \pi_2.\rho_1.r \leq \pi_2.\rho_2.r)
\end{aligned}
$$

Reticulado de permisos (2)

$$\rho_1 = [File \mapsto ((/\text{tmp}/*, \{\text{read}, \text{write}\}), 1)]$$

$$\rho_2 = [File \mapsto ((*/\text{dupont}/*, \{\text{read}\}), \infty)]$$

Operaciones

- Adquisición de permisos

$$consume : Perm_r \longrightarrow Perm \longrightarrow Perm$$

$$consume.p.\rho \quad := \quad \begin{cases} \rho[r \mapsto (p, m-1)] & (\rho.r = (p', m) \wedge p \leq p') \\ \rho[r \mapsto (\bot, m-1)] & \text{(cualquier otro caso)} \end{cases}$$

- Consumo de permisos

$$grant : Perm_r \longrightarrow \mathbb{N} \cup \{\infty\} \longrightarrow Perm \longrightarrow Perm$$

$$grant.(p, m).\rho \quad := \quad \rho[r \mapsto (p, m)]$$

Error

$$\exists rt \in ResType, \exists (p, m) \in Perm_{rt} \times Mul,$$

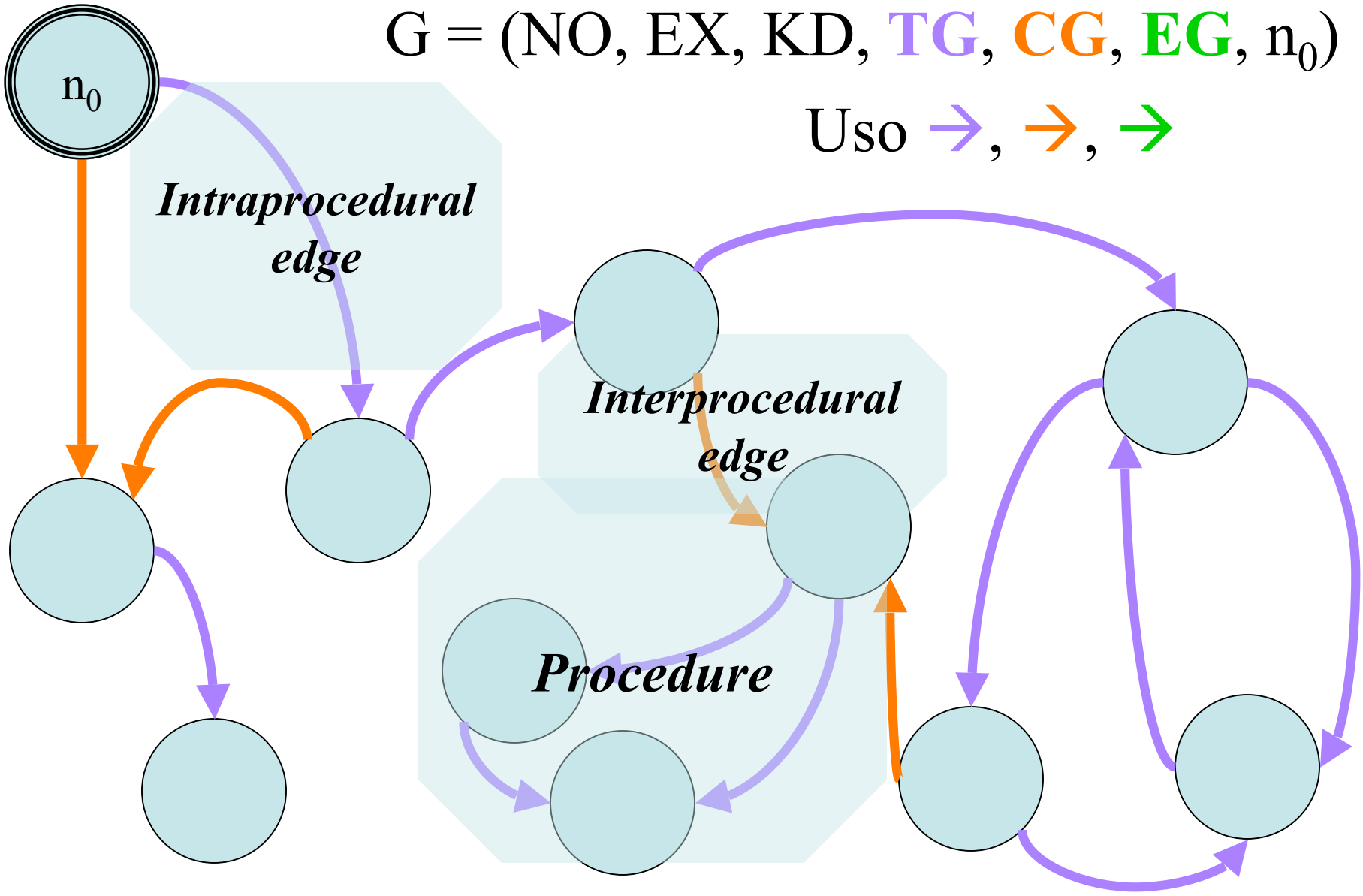
$$\rho(rt) = (p, m) \wedge (p = \bot \vee m = \bot_{Mul})$$

Un programa

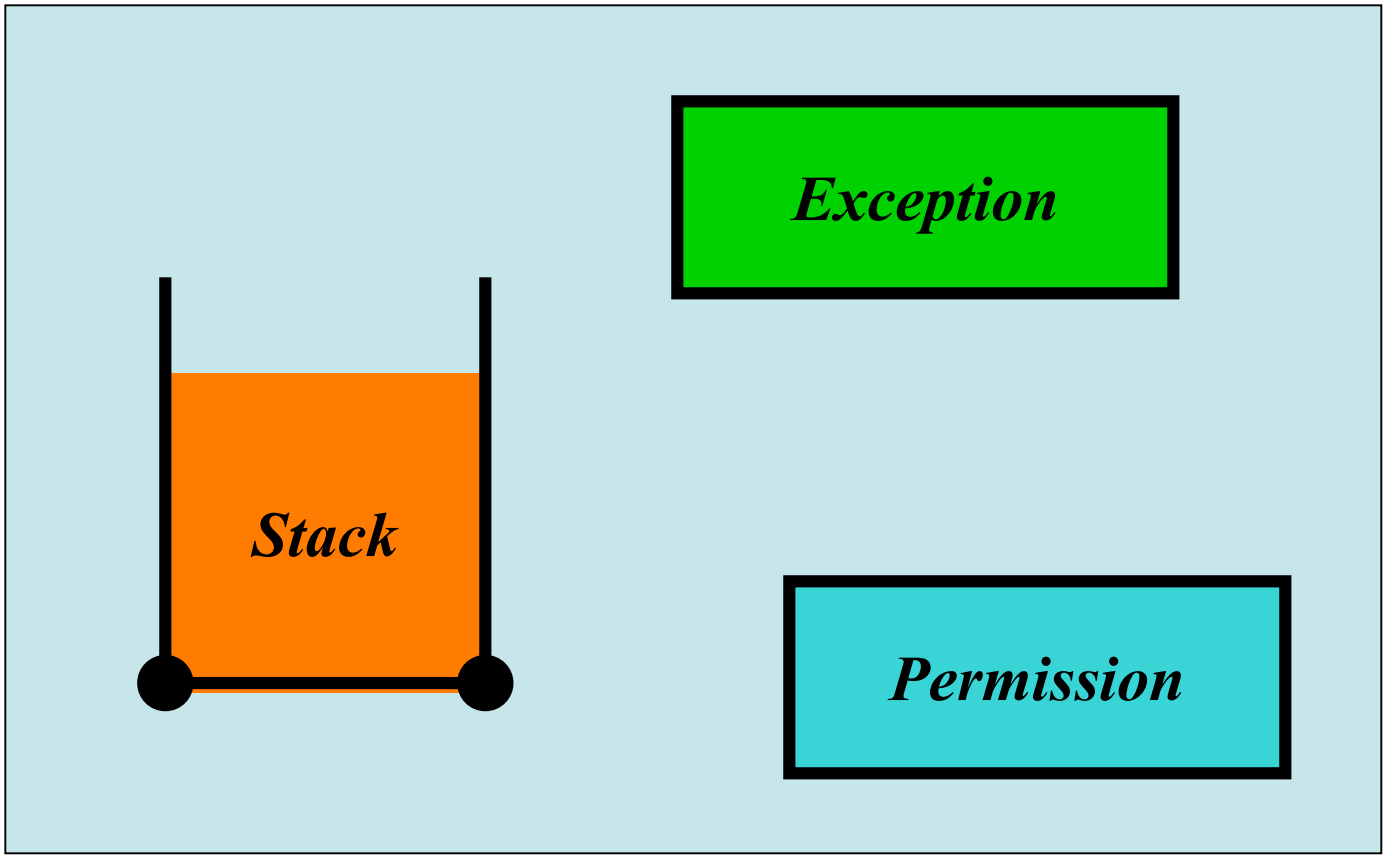
```
grant (http ('*'), read, inf)
grant (https ('site'), read, 1)
grant (file ('walletId'), read, 1)
while True:
    while True: consume (http ('site'), read)
    if True: consume (http ('*'), read)
    else: break
consume (file ('walletId'), read)
if True: consume (http ('site'), read)
else:
    grant (file ('walletVisa'), read, 1)
    consume (file ('walletVisa'), read)
    consume (https ('site'), read)
```

Modelos de programas

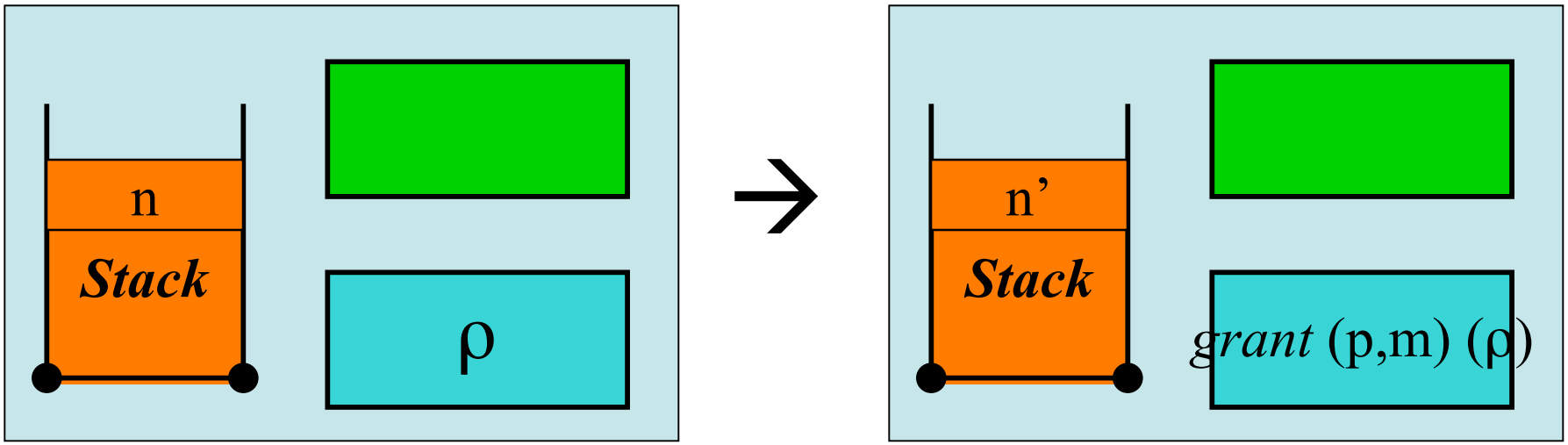
- El programa se modela con un grafo de flujo de control
- Este grafo acepta excepciones y procedimientos


G = (NO, EX, KD, TG, CG, EG, n_0)
Uso →, →, →
n_0
Intraprocedural edge
Interprocedural edge
Procedure

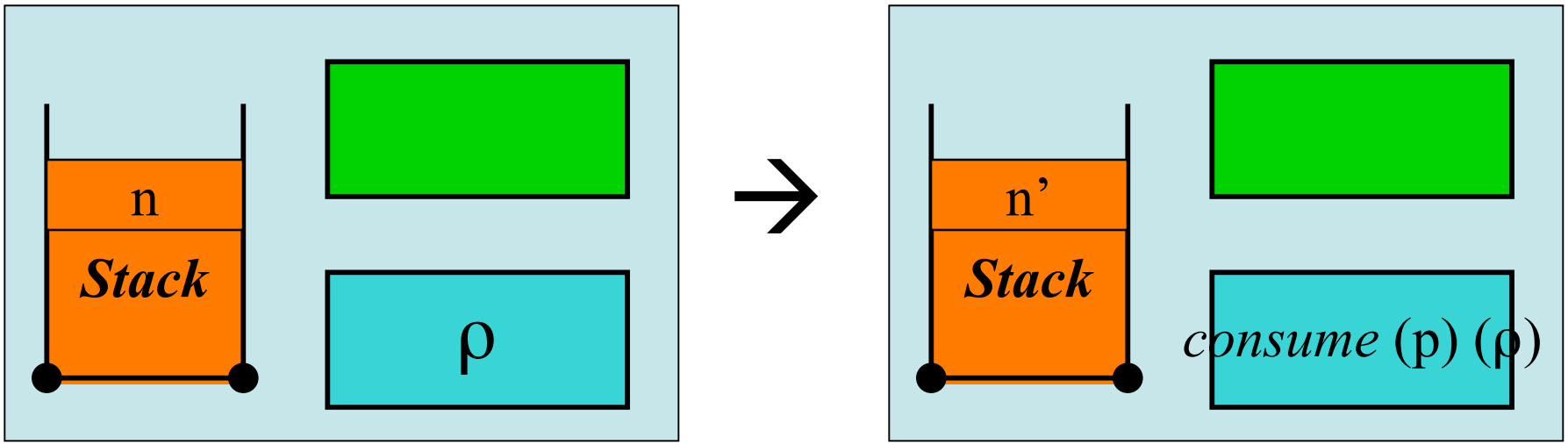
Maquina



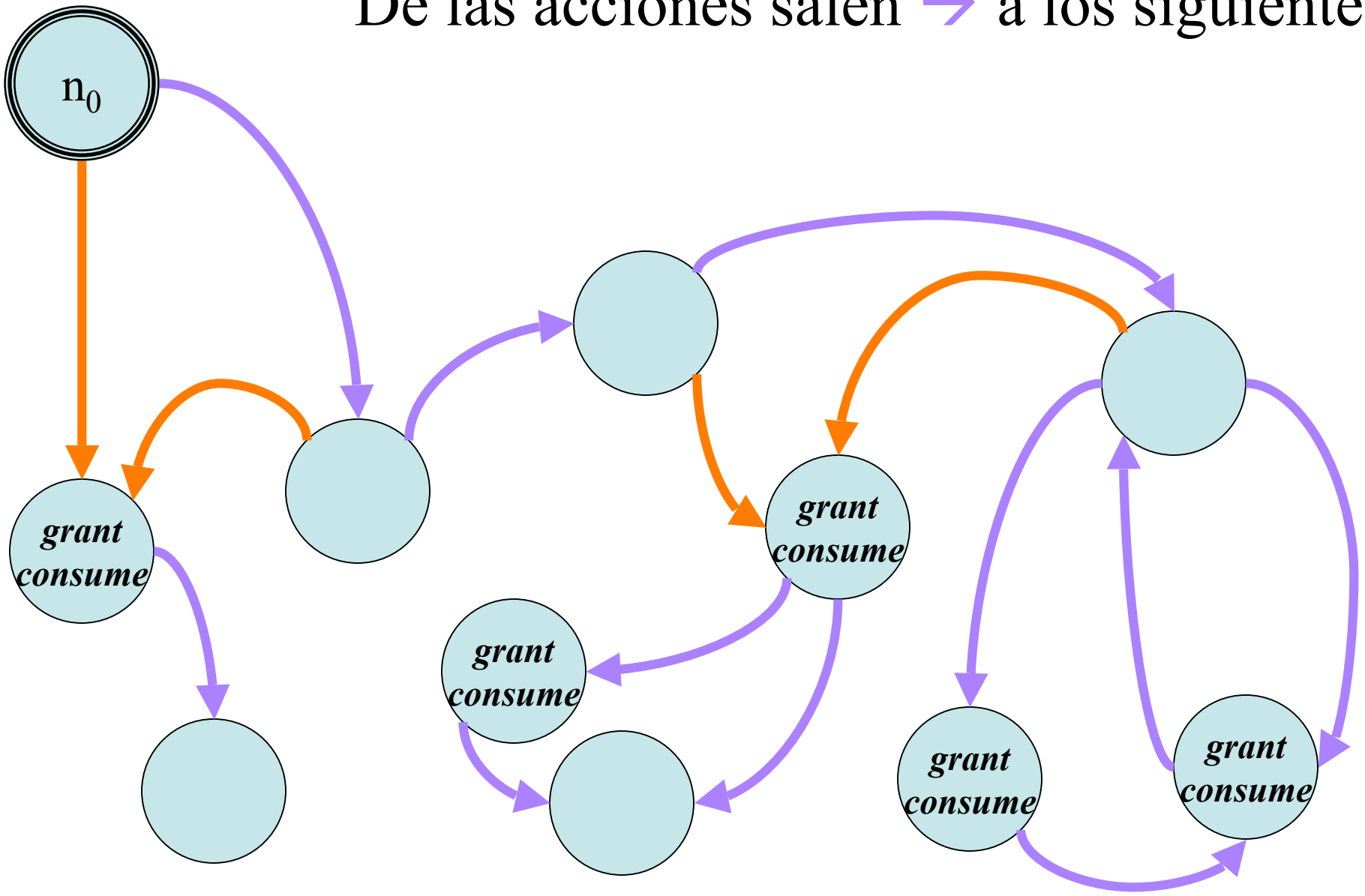
$$\frac{KD(n) = \texttt{grant}\ (p,m) \qquad n \rightarrow n'}{n{:}s,\ \varepsilon,\ \rho \rightarrow n'{:}s,\ \varepsilon,\ \textit{grant}\ (p,m)\ (\rho)}$$



$$\frac{KD(n) = \texttt{consume}\ (p) \qquad n \rightarrow n'}{n{:}s,\ \varepsilon,\ \rho \rightarrow n'{:}s,\ \varepsilon,\ \textit{consume}\ (p)\ (\rho)}$$



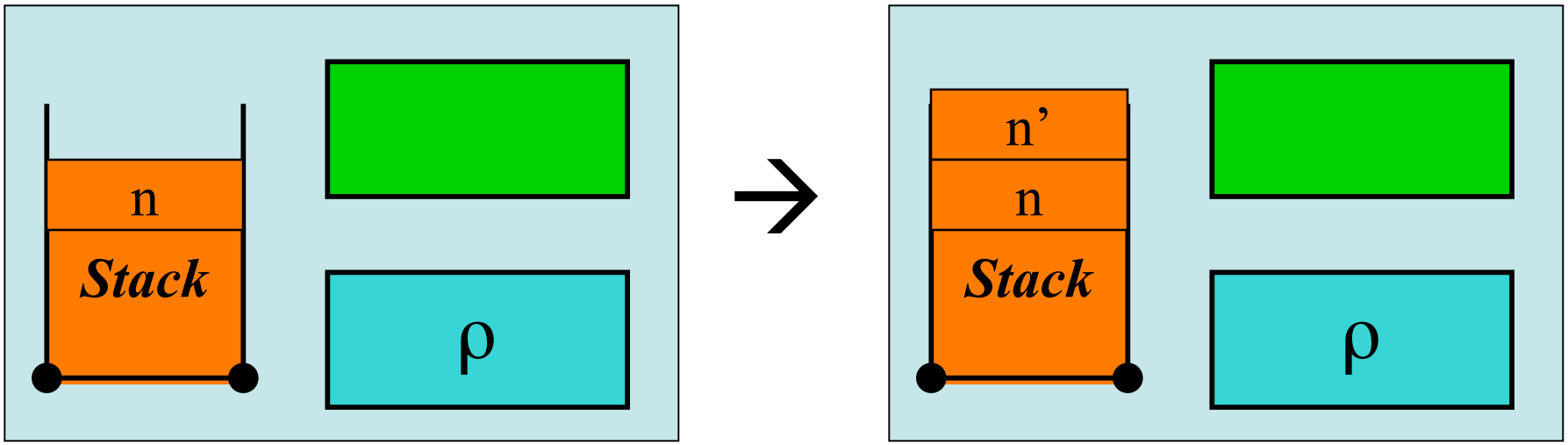
De las acciones salen → a los siguientes



KD(n) = `call` $\quad\quad$ n → n’

---

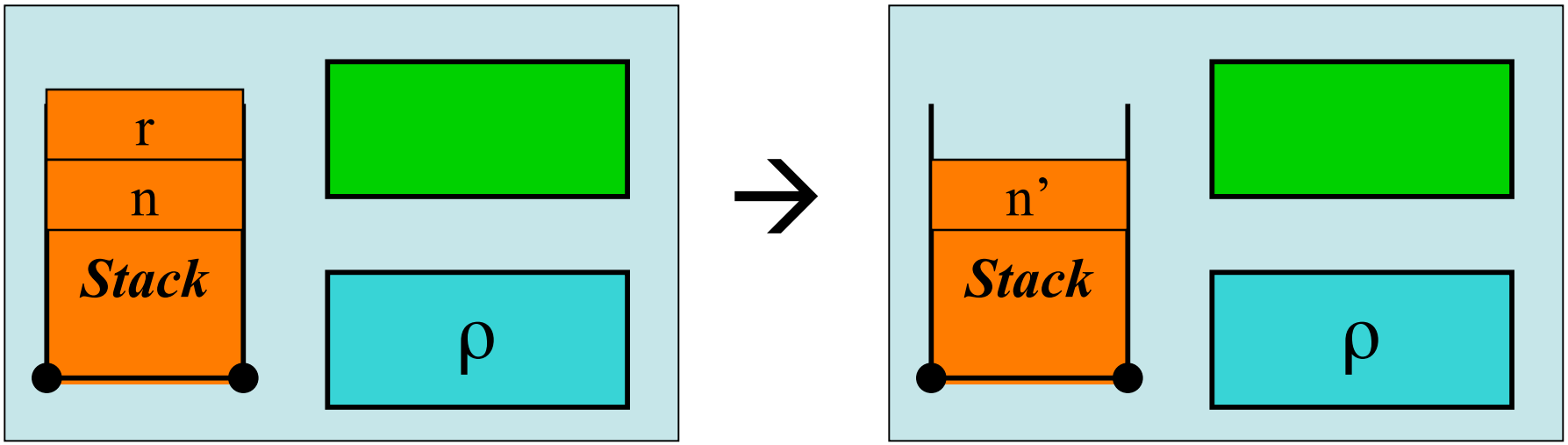
n:s, ε, ρ → n’:n:s, ε, ρ



KD(r) = `return` $\quad\quad$ n → n’

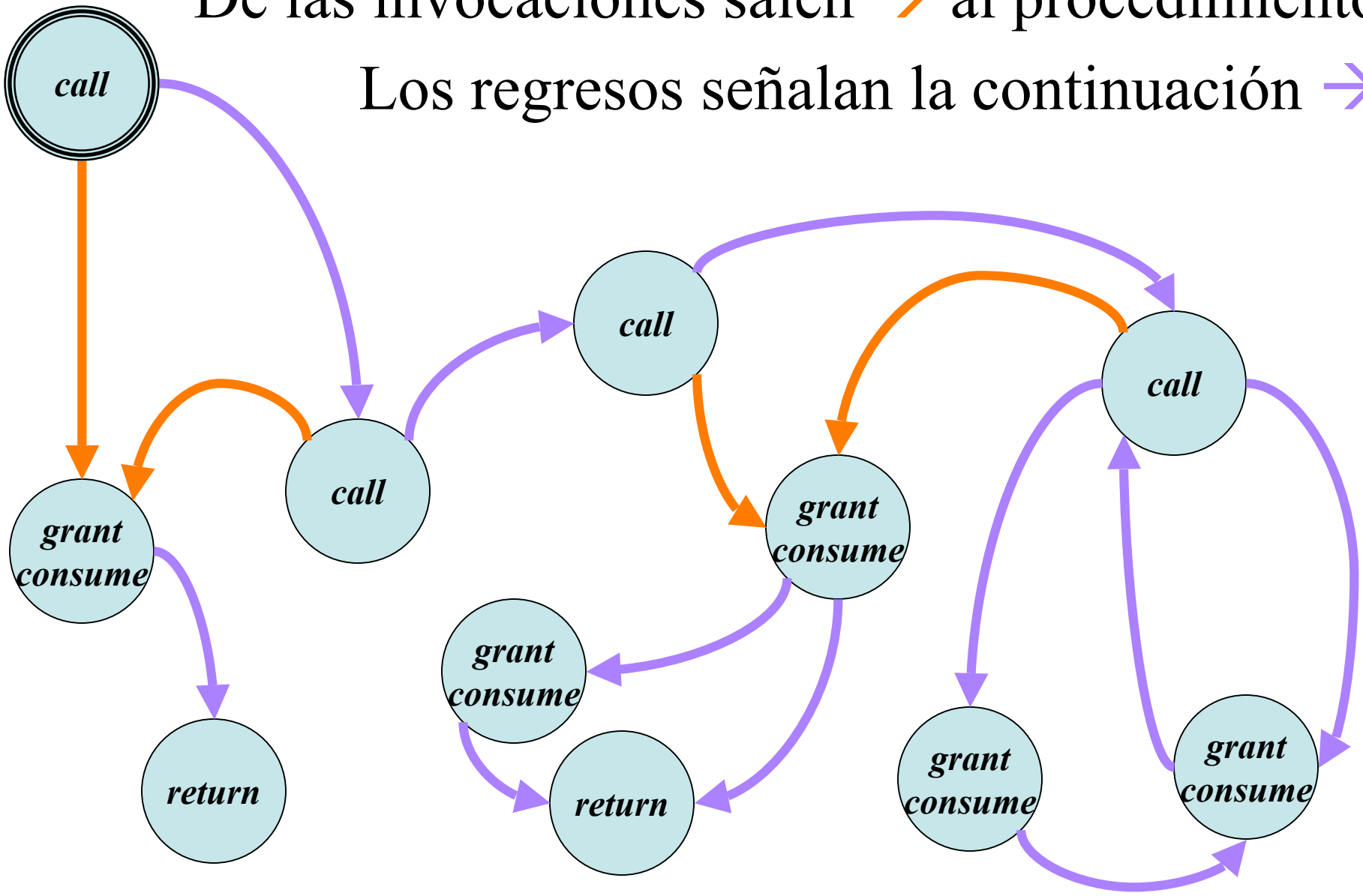
---

r:n:s, ε, ρ → n’:s, ε, ρ



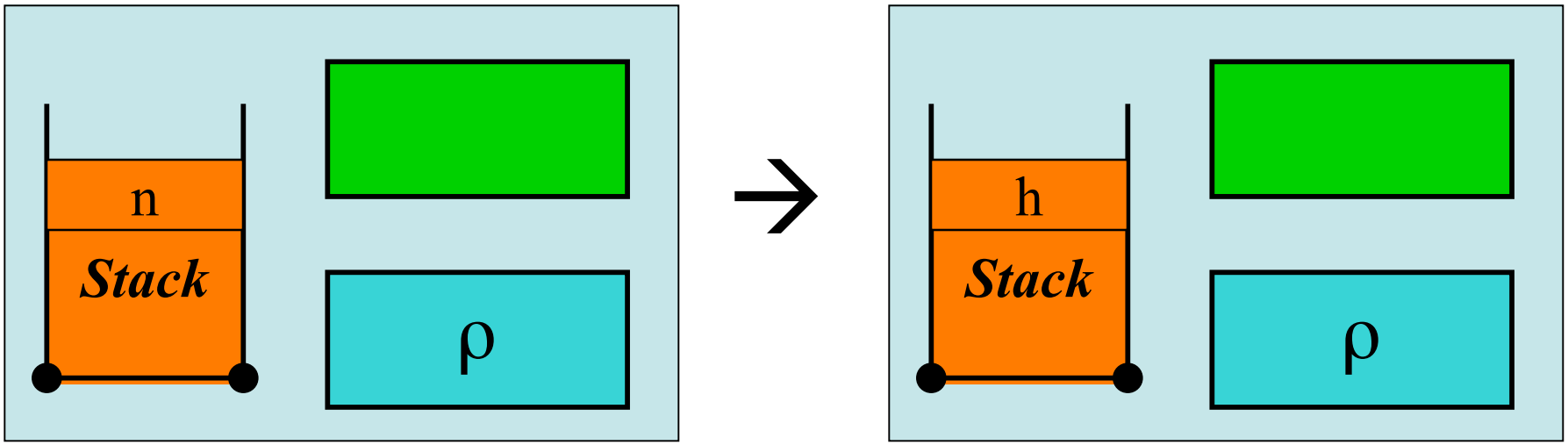
De las invocaciones salen → al procedimiento

Los regresos señalan la continuación →



$$\frac{KD(n) = \texttt{throw}\ (ex)}{n \xrightarrow{ex} h}$$

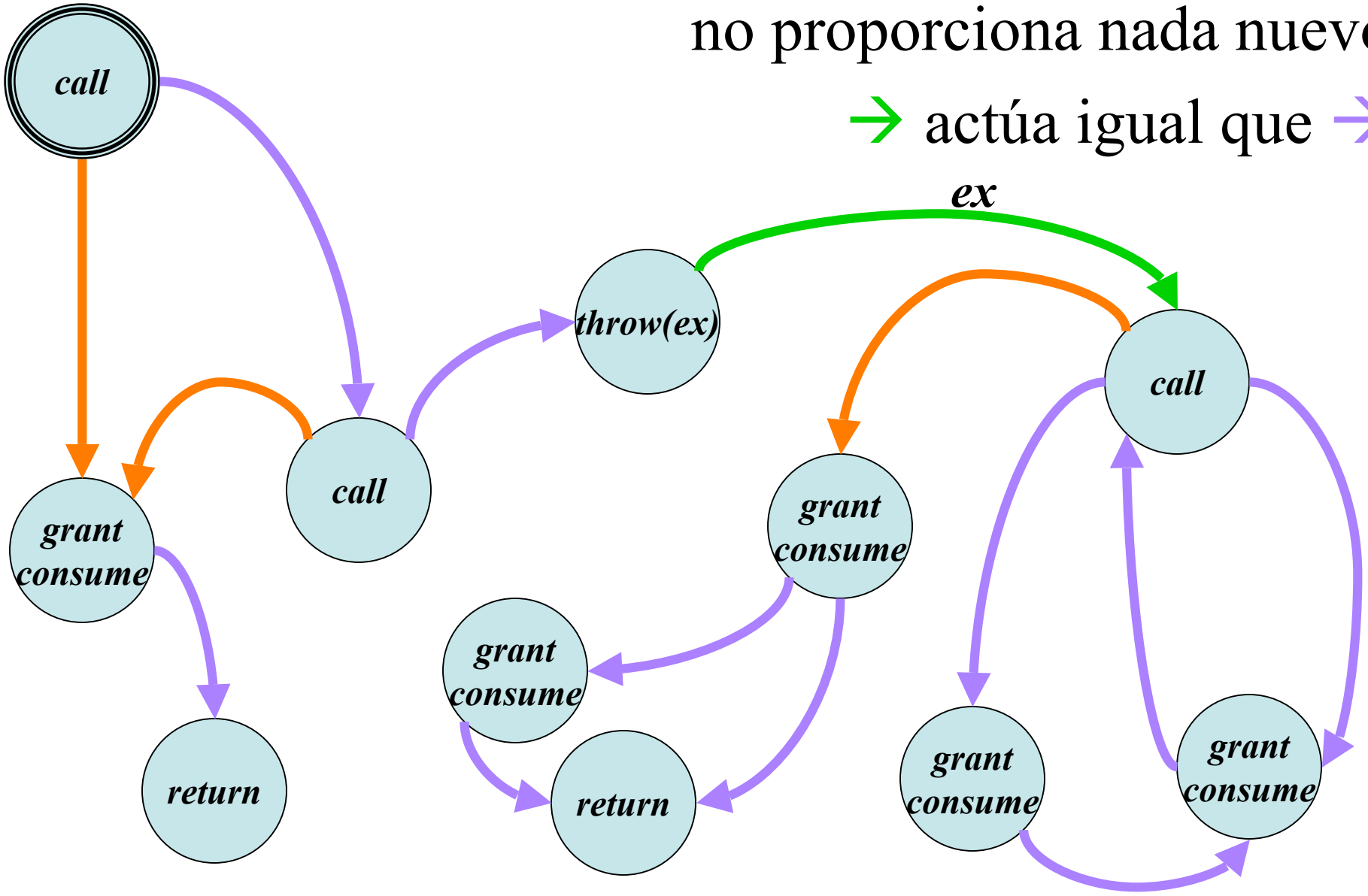
$$n{:}s,\ \varepsilon,\ \rho \rightarrow h{:}s,\ \varepsilon,\ \rho$$



Levantar estas excepciones

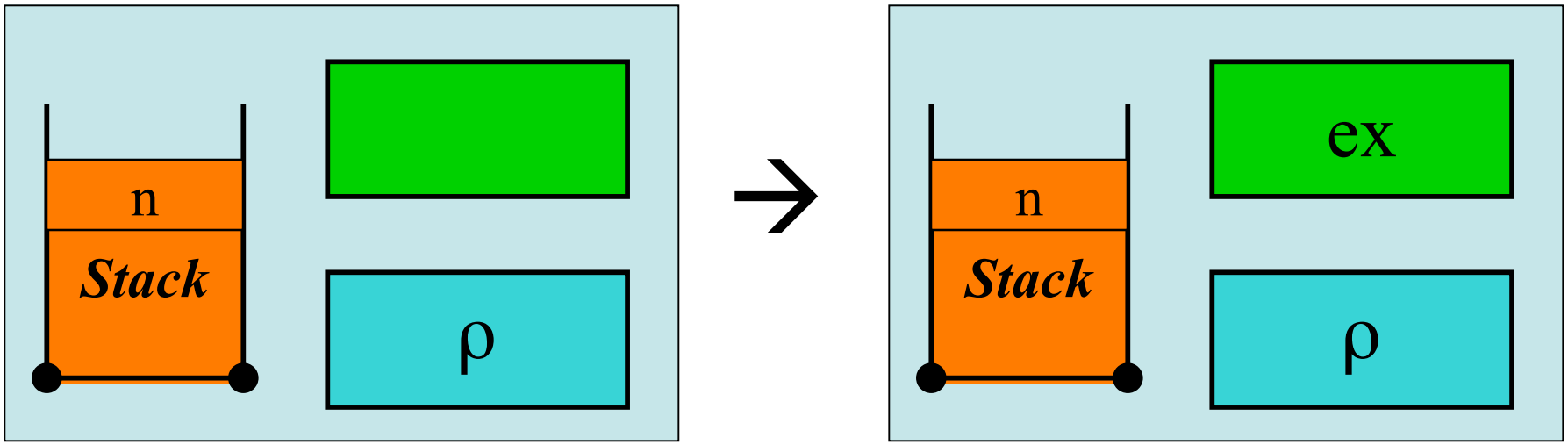
no proporciona nada nuevo

→ actúa igual que →



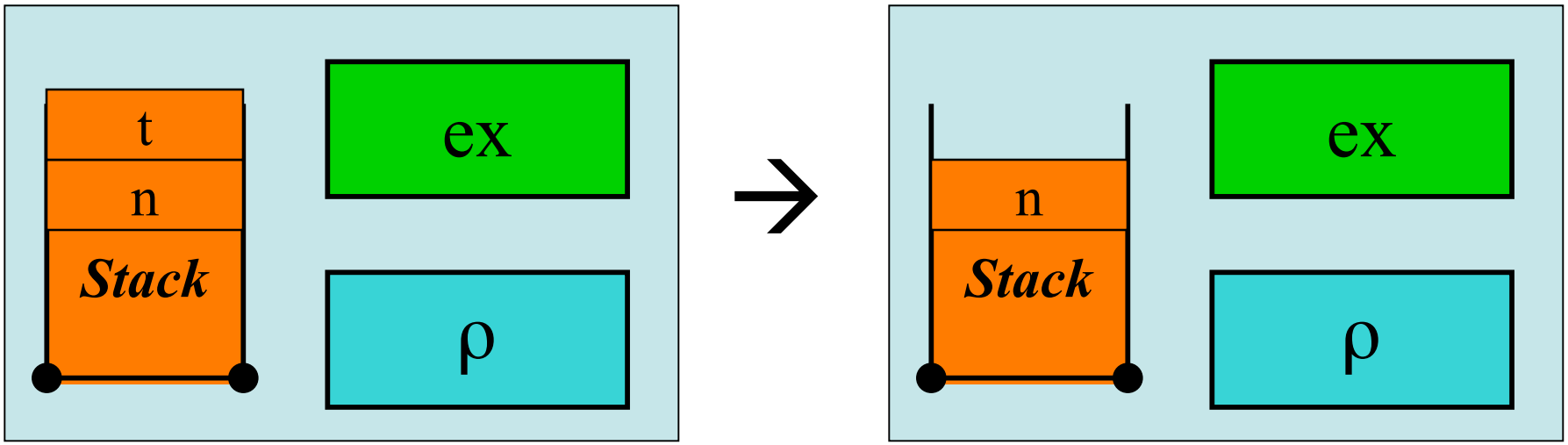
$$KD(n) = \texttt{throw}\ (ex) \qquad \neg(\exists h :: n \xrightarrow{ex} h)$$

$$n{:}s,\ \varepsilon,\ \rho \rightarrow n{:}s,\ ex,\ \rho$$



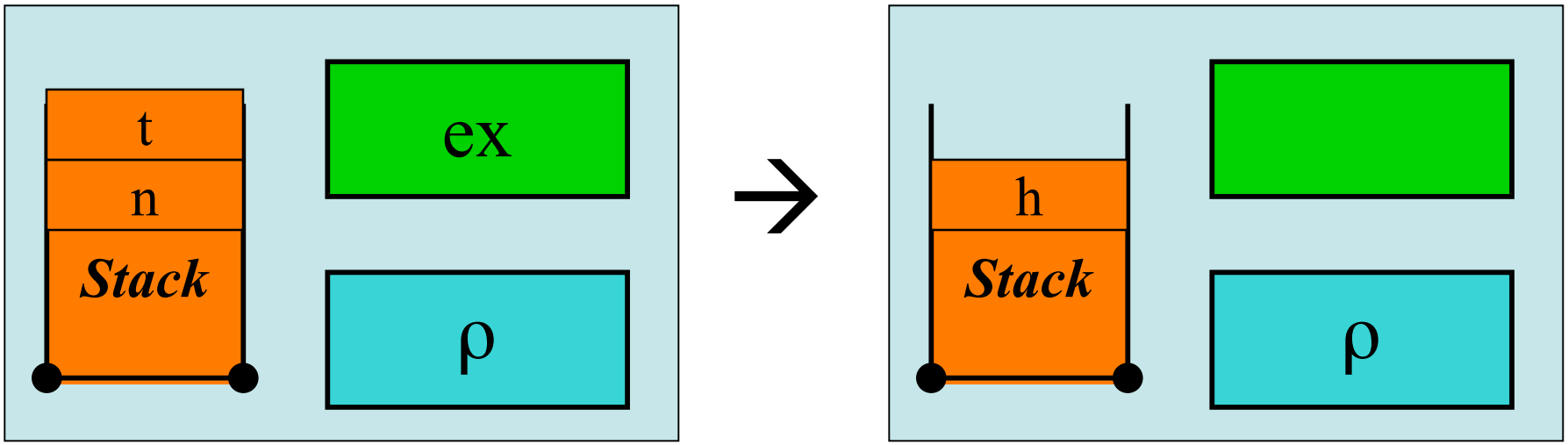
$$\neg(\exists h :: n \rightarrow^{ex} h)$$

$$t{:}n{:}s,\ ex,\ \rho \rightarrow n{:}s,\ ex,\ \rho$$

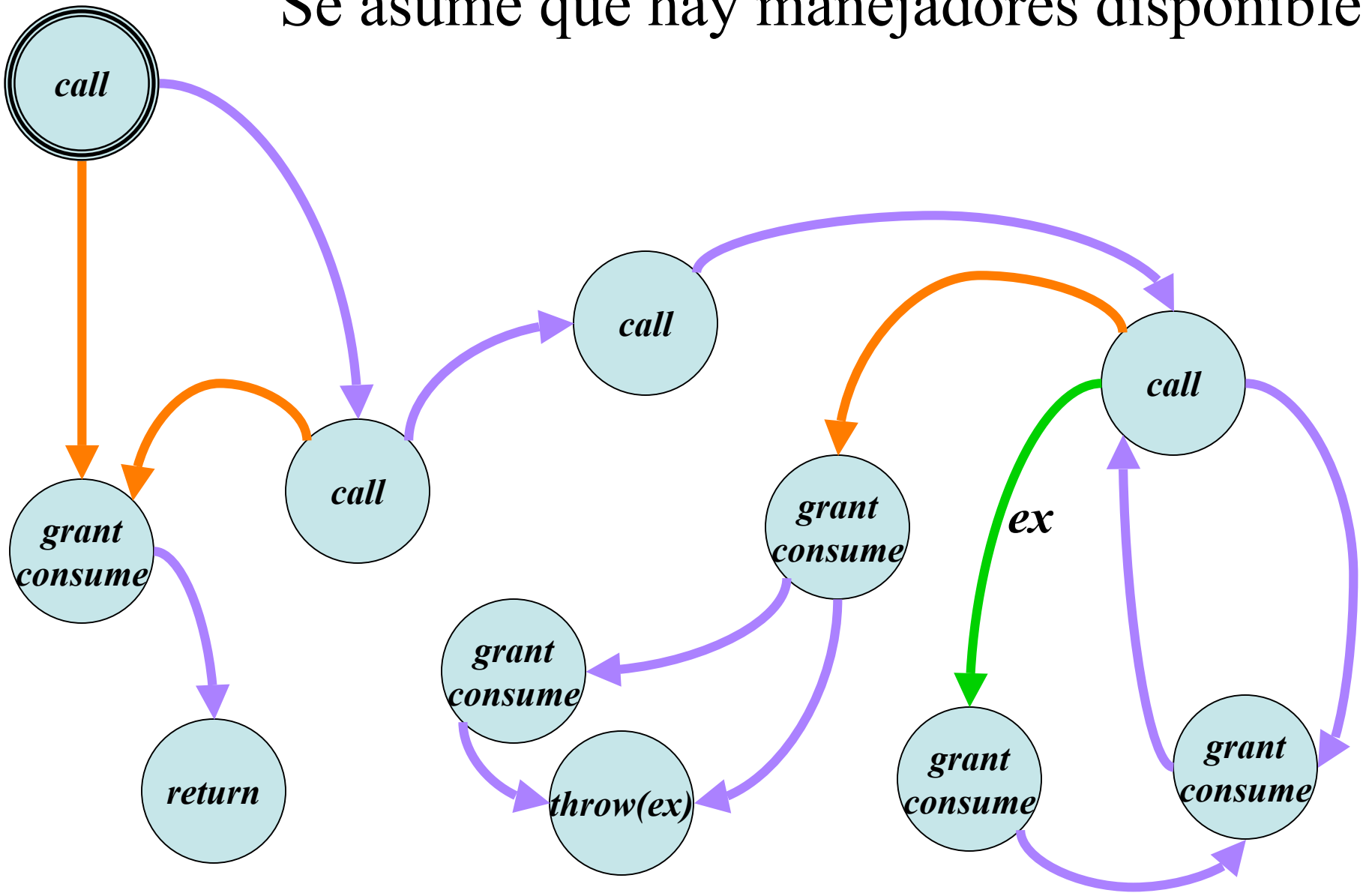


$$n \xrightarrow{ex} h$$

$$t{:}n{:}s,\ ex,\ \rho \rightarrow h{:}s,\ \varepsilon,\ \rho$$



Se asume que hay manejadores disponibles



Uso de los permisos

- La semántica operacional dada no se bloquea por falta de permisos
- El análisis estático propuesto se dirige justamente a esa propiedad: ***el programa nunca intentará acceder a un recurso para el que no tiene permiso***

- Si un programa satisface nuestro análisis, no será necesario verificar esta situación en tiempo de ejecución

Análisis estático

- Técnicas que permiten computaciones aproximadas pero seguras
- Si hay un intento de uso indebido, el análisis lo detecta
- Pero puede suceder que considere usos indebidos cuando no los hay
- ¿Por qué usar un cálculo aproximado?
  - Porque es más barato que un cálculo exacto
  - Porque un cálculo exacto no es posible

Análisis estático

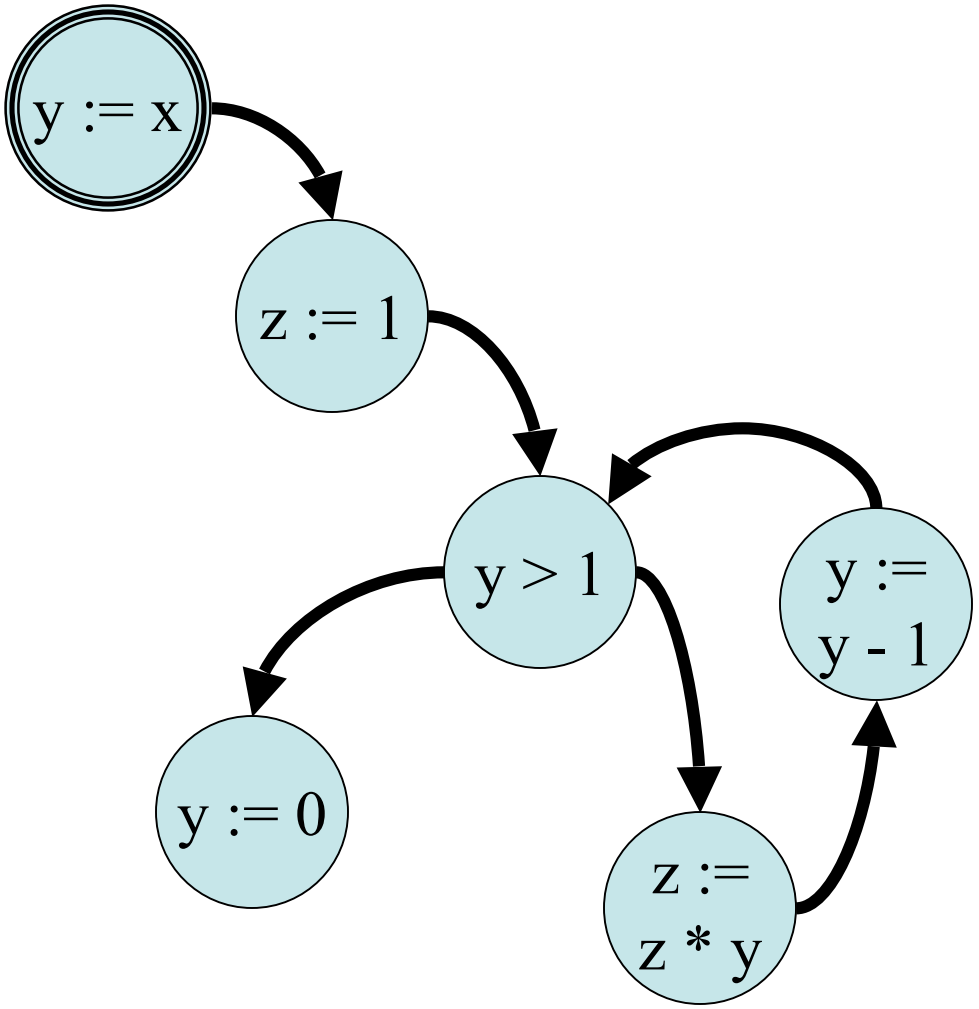
- El programa es un grafo de flujo de control
- Usado en la fase de optimización de los compiladores
- La idea central es
  - recolectar información (que debe tener una estructura de semireticulado)
  - hasta que ya no se pueda obtener más
- Es un cálculo de punto fijo
  - Primo cercano de la verificación de modelos

Ejemplo: reaching definition

- Una asignación $[x := a]^l$ llega a un punto l' del programa si hay alguna ejecución en que la última asignación a x ocurre en l.

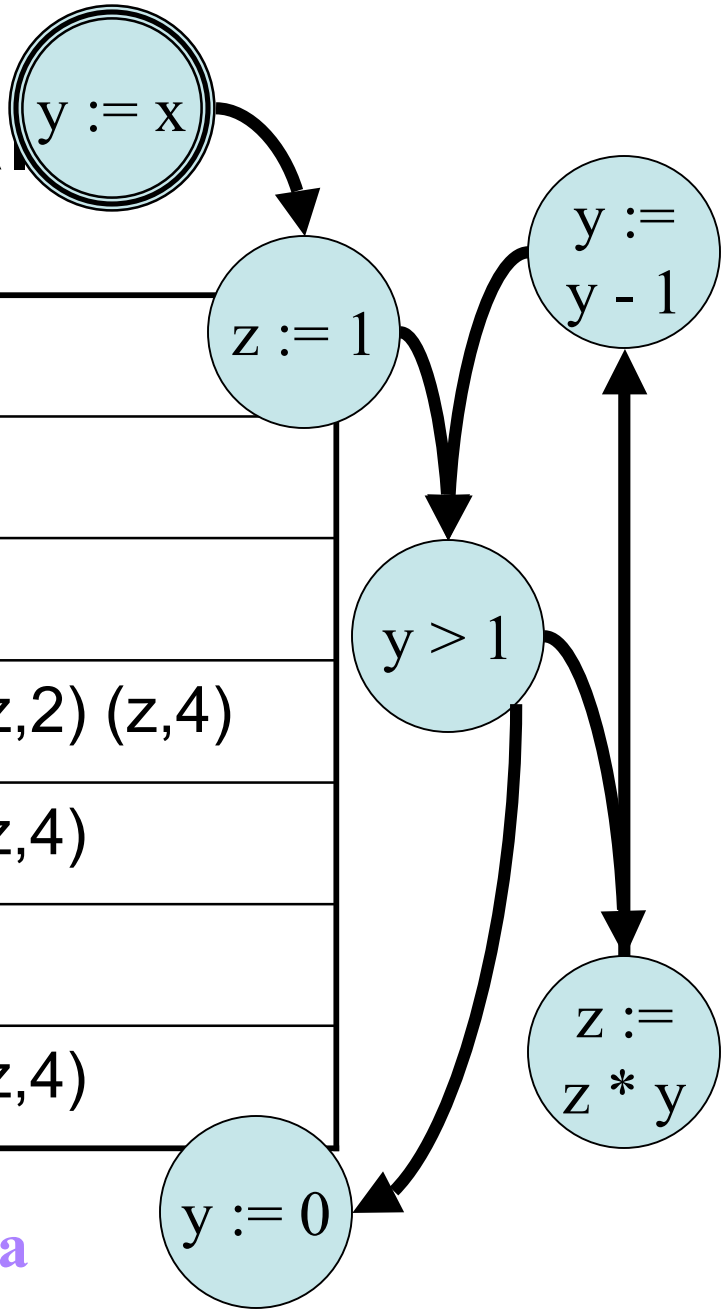
Factorial

```
[y := x]1;
[z := 1]2;
while [y > 1]3 do
    [z := z * y]4;
    [y := y - 1]5;
[y := 0]6
```



Factorial

	entry	exit
1	(y,?) (z,?)	(y,1) (z,?)
2	(y,1) (z,?)	(y,1) (z,2)
3	(y,1) (y,5) (z,2) (z,4)	(y,1) (y,5) (z,2) (z,4)
4	(y,1) (y,5) (z,2) (z,4)	(y,1) (y,5) (z,4)
5	(y,1) (y,5) (z,4) (z,2)	(y,5) (z,4)
6	(y,1) (y,5) (z,2) (z,4)	(y,6) (z,2) (z,4)



Información segura, pero menos exacta

Información insegura (el cálculo es inconsistente)

El programa

```
def MFP (a):
    W = WLmap [WLoption] (a.flow)
    for (l1, l2) in W.iter ():
        fl = a.transfer (l1)
        if (not a.latt.leq (fl, a.a [l2])):
            a.a [l2] =
                a.latt.join (a.a [l2], fl)
            W.add ([(s,t)
                for (s,t) in a.flow if s==l2])
    a.dump ()
```

Un análisis
extrae información de un programa
mediante una función de transferencia

Trabaja sobre un reticulado
iterando la búsqueda de nueva información.

Ejemplo: live variables

```
class Analysis (MF):
   def init_(self, pgm): MF.init_(self, pgm, 'BW')
   def defLattice (self): self.latt = SetVarLat ()
   def defextremalValue (self):
      self.extremalValue = SetVar ([])
   def transfer (self, l):
      return SetVar.union (\
         SetVar.diff (self.a[l], self.kill (l)),\
         self.gen(l))
   def kill (self, l):
      return SetVar (eval (getKill(self.Blocks[l])))
   def gen (self, l):
      return SetVar (eval (getGen (self.Blocks[l])))
```

Análisis estático: uso de permisos

- El valor a calcular (aproximadamente) son los permisos P_n que están garantizados en el punto de programa n

- Un programa será correcto si

$$(\forall n : \text{not Error } (P_n))$$

$$P_{n0} \subseteq p_{init}$$

$$\frac{KD(n) = \texttt{consume}\ (p) \qquad\qquad n \rightarrow m}{P_m \subseteq \mathit{consume}\ (p)\ (P_n)}$$

$$\frac{KD(n) = \texttt{grant}\ (p, c) \qquad\qquad n \rightarrow m}{P_m \subseteq \mathit{grant}\ (p, c)\ (P_n)}$$

$KD(n) = \texttt{call}$ $\quad\quad n \rightarrow m$

$$P_m \subseteq P_n$$

$$KD(n) = \texttt{throw}\ (ex) \qquad n \xrightarrow{ex} m$$

$$P_m \subseteq P_n$$

$$\frac{KD(n) = \texttt{call} \qquad n \rightarrow b \qquad n \rightarrow m}{P_m \subseteq R_b(P_n)}$$

$$\frac{KD(n) = \texttt{call} \qquad n \rightarrow b \qquad n \rightarrow^{ex} m}{P_m \subseteq R^{ex}_b(P_n)}$$

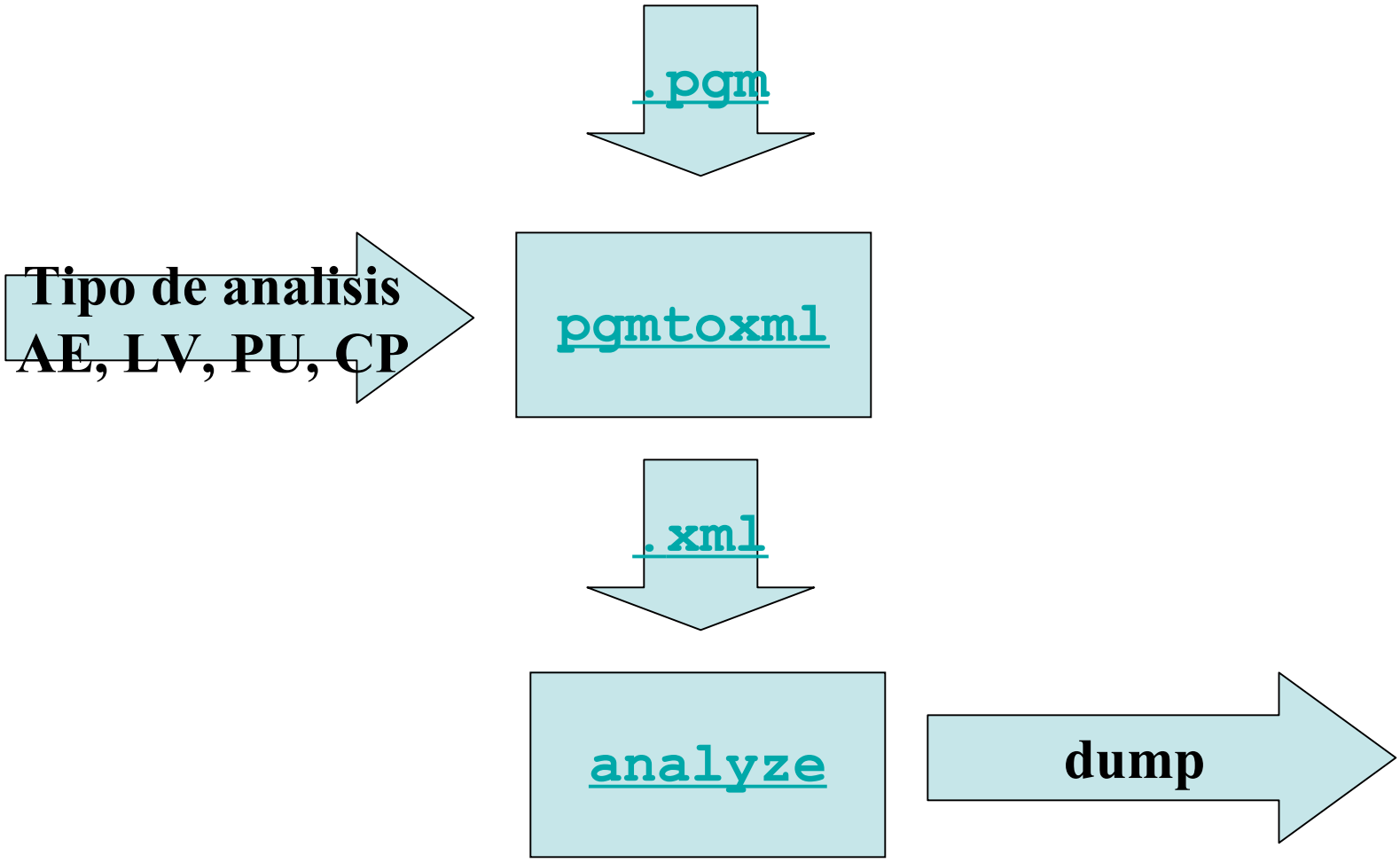
$$KD(n) = \texttt{call} \qquad n \rightarrow b \qquad \neg(\exists h :: n \rightarrow^{ex} h)$$

$$P_m \subseteq R^{ex}_b(P_n)$$

Código

```
class Analysis (MF):
   def __init__ (self, pgm): MF.__init__ (self, pgm, 'FW')
   def evalMeta (self):
      MF.evalMeta (self)
      self.Action = eval (self.meta.getAttribute ('Actions'))
      self.ResType = eval (self.meta.getAttribute ('ResType'))
      self.Resources = eval (self.meta.getAttribute ('Resources'))
   def defLattice (self):
      self.latt = Perm (self.Resources, self.Action, self.ResType)
   def defextremalValue (self): self.extremalValue = self.latt.bottom ()
   def transfer (self, l):
      t = deepcopy (self.a[l])
      sk, sg = self.kill (l), self.gen (l)
      if sk:
         rt, res, act = sk.split ()
         t.consume (rt, SetStr (set ([res])), SetStr (set ([act])))
      if sg:
         rt, res, act, m = sg.split ()
         if m != 'inf': m = int (m)
         t.grant (rt, SetStr (set ([res])), SetStr(set ([act])), m)
      return t
```

Programa



```
grant (http ('*'), read, inf)
grant (https ('site'), read, 1)
grant (file ('walletId'), read, 1)
while True:
    cor
    if
    els
```

```
<?xml version="1.0" ?>
<pgm>
 <meta Actions="set(['read'])" Label="15" ResType="set(['http',
'file', 'https'])" Resources="set(['walletId', 'walletVisa',
'site'])"/>
 <main>
   <command gen="http * read inf" kill="" label="1"/>
   <command gen="https site read 1" kill="" label="2"/>
   <command gen="file walletId read 1" kill="" label="3"/>
   <loop breaks="[9]" label="4">
     <loop breaks="[]" label="5">
    ••••••••
   </branch>
 </main>
</pgm>
```

1 → P1
2 → P2
3 → P3
4 → ERROR
5 → P4
••••••••

Situación actual

- Tenemos una propuesta de modelo de permisos para MIDP
- Tenemos una idea de cómo llevar esa propuesta a Coq
- Tenemos una implementación (aún parcial) de un analizador estático

Otras tareas

- Estudiar en qué medida están relacionados las realizaciones del modelo de permisos
  - Coq versus Python
  - El gallo y la serpiente
- Proponer modificaciones al modelo de permisos

Y luego …

- Formalizar en Coq el problema del análisis estático, que va de la mano de formalizar reticulados
- Modelar formalmente otras propuestas de seguridad
- Buscar puntos de encuentro entre el modelado formal y el desarrollo de software

He robado a …

- *A Formal Model of Access Control for Mobile Interactive Devices.* Frédéric Besson, Guillaume Dufay, and Thomas Jensen
- *Embedded Java Security Security for Mobile Devices.* Mourad Debbabi, Mohamed Saleh, Chamseddine Talhi and Sami Zhioua