Tema 3: Planificación Inteligente

Sistemas Inteligentes
3 Grado de Ingeniería Informática
Esp. Computación

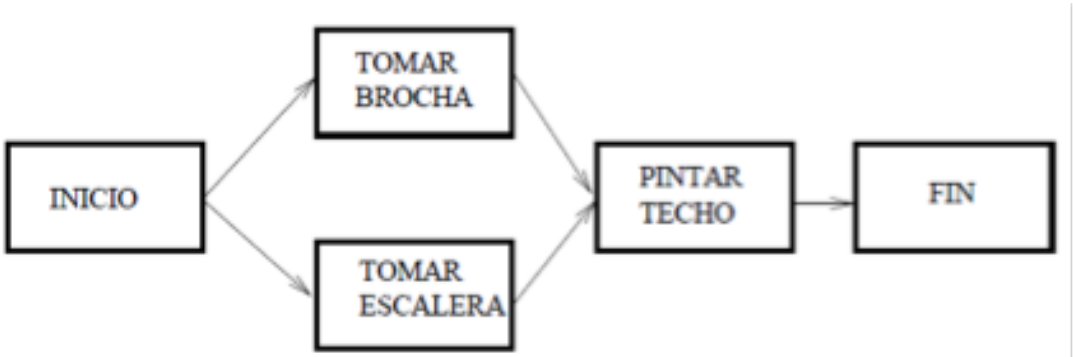
Planificación de Orden Parcial

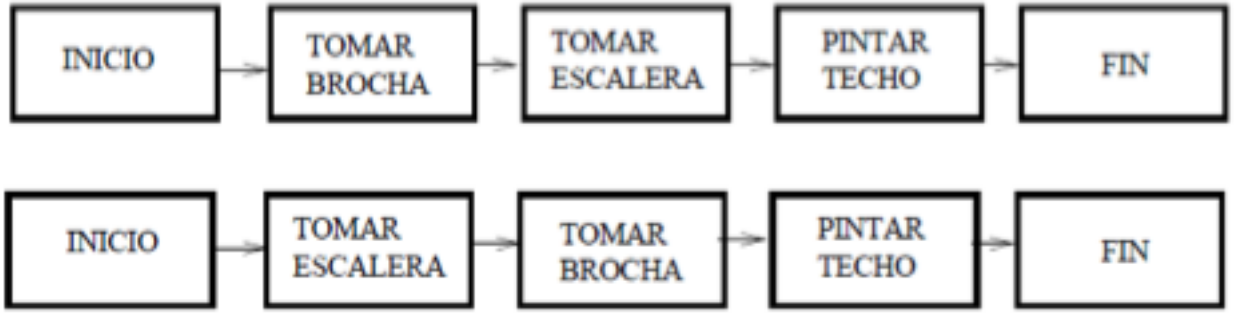
Planes ordenados parcialmente

- Plan parcialmente ordenado:
  - un plan en el que sólo se especifican algunas de las precedencias entre sus acciones
- Principio de mínimo compromiso
- Secuenciar un plan parcialmente ordenado: construir un plan secuencial que respete las precedencias obligatorias
  - Este será el último paso

Planes ordenados parcialmente

- Plan parcialmente ordenado y secuenciaciones





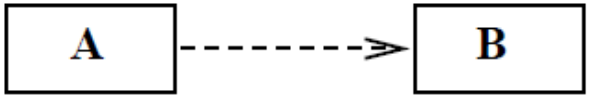
Componentes

- Un conjunto de acciones que constituyen los pasos que el plan lleva a cabo, de entre los operadores del problema, cada una con sus precondiciones y efectos

- Dos acciones especiales:
  - INICIO: (sin precondiciones y cuyo efecto es el estado inicial) y
  - FIN: (sin efectos y cuyas precondiciones son el objetivo final)

Componentes

- Un conjunto de restricciones de orden A $\prec$ B entre acciones del plan

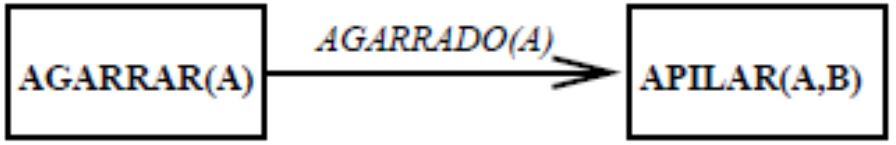


Componentes

- Un conjunto de enlaces causales

$$A \xrightarrow{p} B$$

- especificando la consecución, por parte de una acción, de una de las precondiciones de otra acción del plan (lleva implícita una restricción de orden)



- Un conjunto de precondiciones abiertas: aquellas que aún no tienen enlaces causales que las consigan

Planes parciales inicial y final

- Plan parcial inicial:
  - plan cuyas únicas acciones son INICIO y FIN, con la restricción INICIO $\prec$ FIN, sin enlaces causales y con todas las precondiciones de FIN abiertas

- Planes parciales finales:
  - planes parciales sin conflictos* entre los enlaces causales, sin ciclos entre las restricciones de orden y sin precondiciones abiertas

  - *Una acción C entra en conflicto con (o amenaza) un enlace causal A $\rightarrow p \rightarrow$ B, si C tiene a $\neg p$ en su lista de efectos y según las restricciones de orden, C podría ir después de A y antes que B*

Ejemplo: Aprobar SInt.

- Lenguaje:
  - Objetos: SInt, ETSI, CASA
  - Predicado: EN(-), ESTUDIADO(-), APROBADO(-)
- Estado inicial:
  - EN(CASA)
- Estado final:
  - EN(CASA), APROBADO(SInt)

Ejemplo: Aprobar(SInt)

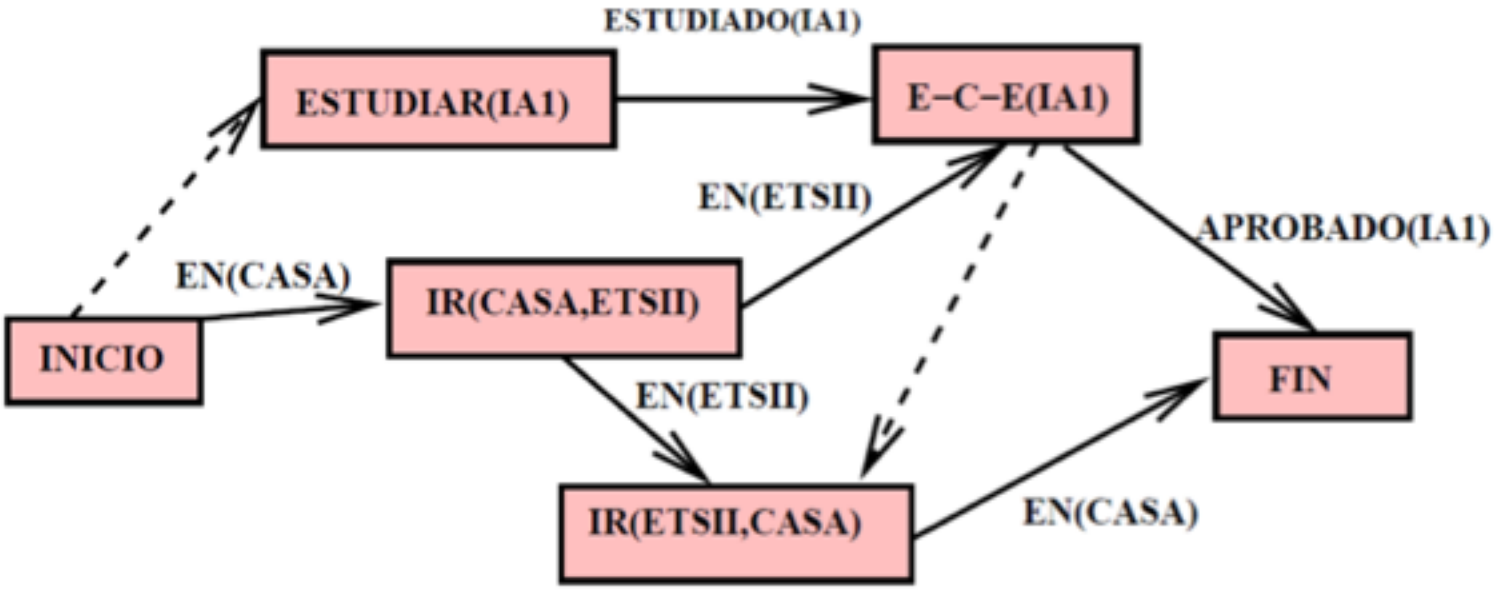
- Acciones:

IR(x,y) :	**ESTUDIAR(x):**	**EXAMINARSE-CON-EXITO(x):**
P: EN(x) B: EN(x) A: EN(y)	P: {} B: {} A: ESTUDIADO(x)	P: EN(ETSI), ESTUDIADO(x) B: {} A: APROBADO(x)

Descartaremos acciones del tipo IR(x,x)

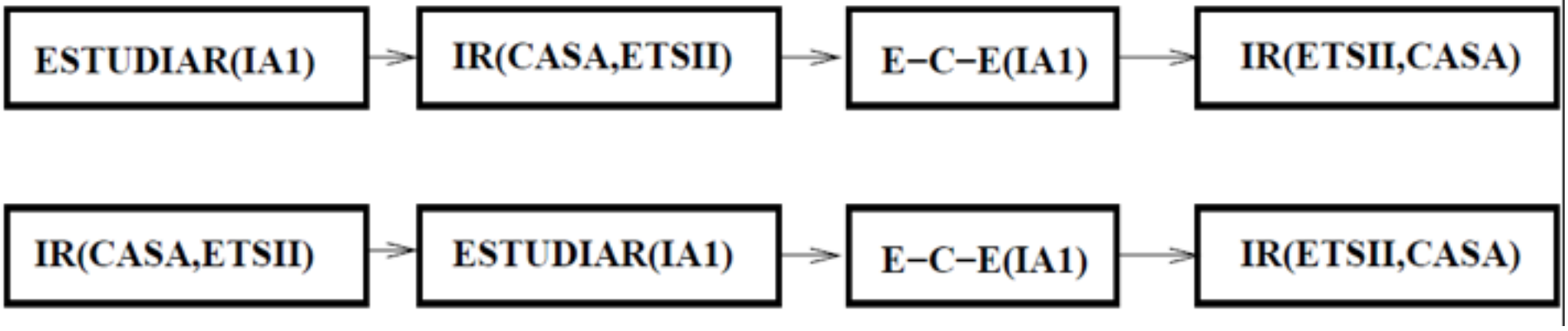
Ejemplo: Aprobar(SInt)

- Ejemplo de plan parcial final:



Ejemplo: Aprobar(SInt)

- Secuenciación de un plan parcial: colocar las acciones una detrás de otra, sin contradecir ninguna restricción de orden que se deduzca del plan parcial
- Punto clave: cualquier secuenciación de un plan parcial solución da lugar a una secuencia de acciones que partiendo del estado inicial consigue el objetivo (es decir, una solución al problema original)
- En el ejemplo, dos posibilidades (ambas soluciones al problema original):



Algoritmo

- ¿Cómo diseñar un algoritmo para encontrar planes parciales finales?
  - Idea: comenzar en el plan inicial y aplicar transformaciones u *operadores* a los planes parciales, refinándolos
  - Básicamente, estas transformaciones consisten en resolver precondiciones abiertas o resolver amenazas
- En cada punto habrá varias alternativas de refinamiento, y no todas conducen hacia un plan parcial final
- Problema: encontrar la secuencia de pasos de refinamiento que partiendo del plan parcial inicial, llegue a un plan parcial solución (es decir, sin ciclos, sin amenazas y sin precondiciones abiertas)
  - ¡Es una búsqueda en espacios de estados!
- Pero ahora *los estados son los planes parciales*

Algoritmo

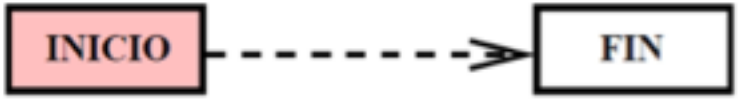
- Resolución de precondiciones abiertas: dada una precondición abierta p de una acción B del plan, por cada acción A que tiene a p como efecto, se puede obtener un plan sucesor (*refinar*) aplicando alguno de los siguientes pasos (siempre que el plan resultante no tenga ciclos)
  - ***Establecimiento simple***: si la acción A ya aparece en el plan, añadir la restricción $A \prec B$ y el enlace causal $A \xrightarrow{p} B$
  - ***Acción nueva***: si la acción A no aparece ya en el plan, añadir la acción al plan, las restricciones $A \prec B$, INICIO $\prec A$ y $A \prec$ FIN, y el enlace causal $A \xrightarrow{p} B$ (es posible incluso introducir como nueva una acción igual a una que ya estuviera)

Algoritmo

- Resolución de conflictos: dado un conflicto entre el enlace causal $A \xrightarrow{p} B$ y la acción C, se puede obtener un plan sucesor aplicando alguno de los siguientes pasos (siempre que el plan resultante no tenga ciclos):
  - Añadir la restricción $B \prec C$ (*promoción*)
  - O añadir la restricción $C \prec A$ (*degradación*)

Ejemplo: Aprobar(SInt)

- Paso 1: plan inicial



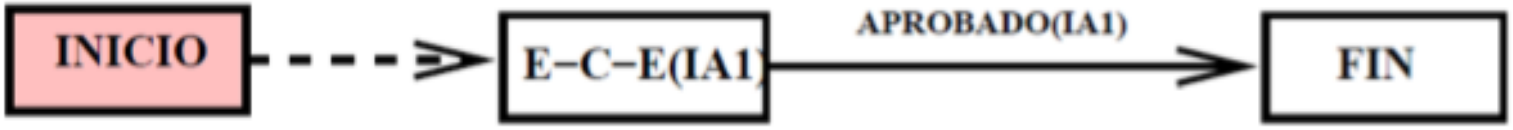
  - Precondiciones abiertas: `APROBADO(IA1)`, `EN(CASA)`; vamos a resolver primero `APROBADO(IA1)`

- Una única posibilidad: acción nueva EXAMINARSE-CON-EXITO(IA1)

```
IR(x,y)          ESTUDIAR(x)          EXAMINARSE-CON-EXITO(x)
P: EN(x)           P: {}                P: EN(ETSII), ESTUDIADO(x)
B: EN(x)           B: {}                B: {}
A: EN(y)           A: ESTUDIADO(x)      A: APROBADO(x)
```

Ejemplo: aprobar IAIC

- Paso 2:

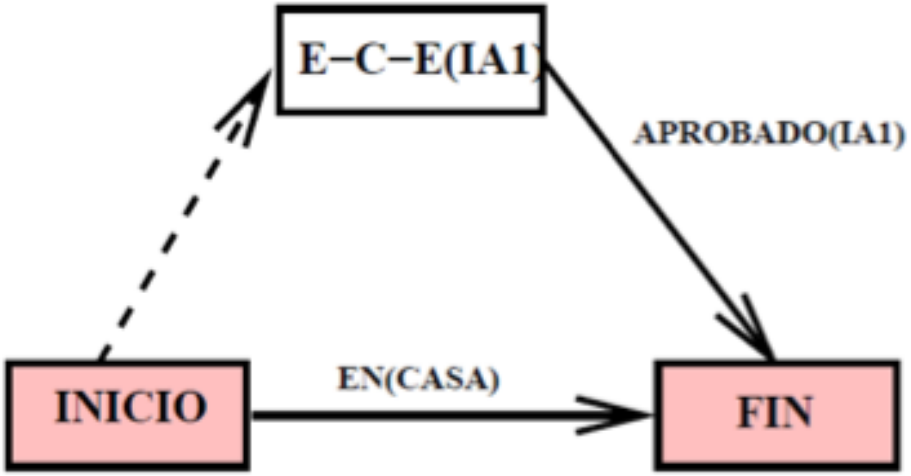


  - Resolvemos ahora la precondición EN(CASA) de FIN
  - Dos posibilidades: establecimiento simple con INICIO ó acción nueva IR(ETSII,CASA)
  - Consideraremos la primera de ella en primer lugar, intentando la segunda en caso de fallo

```
IR(x,y)          ESTUDIAR(x)          EXAMINARSE-CON-EXITO(x)
P: EN(x)           P: {}                P: EN(ETSII), ESTUDIADO(x)
B: EN(x)           B: {}                B: {}
A: EN(y)           A: ESTUDIADO(x)      A: APROBADO(x)
```

Ejemplo: aprobar IAIC

- Paso 3:

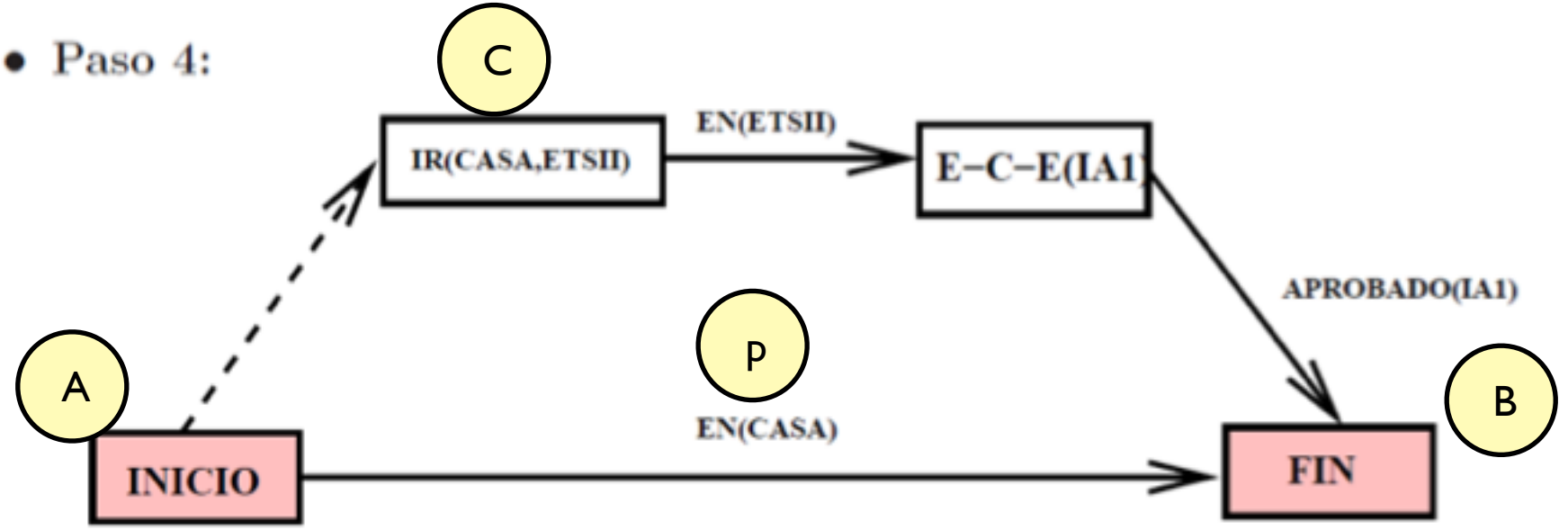


- Resolvemos ahora la precondición EN(ETSII) de EXAMINRSE-CON-EXITO(IA1)
- Sólo una posibilidad: acción nueva IR(CASA,ETSII)

```
IR(x,y)          ESTUDIAR(x)          EXAMINARSE-CON-EXITO(x)
P: EN(x)           P: {}                P: EN(ETSII), ESTUDIADO(x)
B: EN(x)           B: {}                B: {}
A: EN(y)           A: ESTUDIADO(x)      A: APROBADO(x)
```

Ejemplo: aprobar IAIC

- Paso 4:

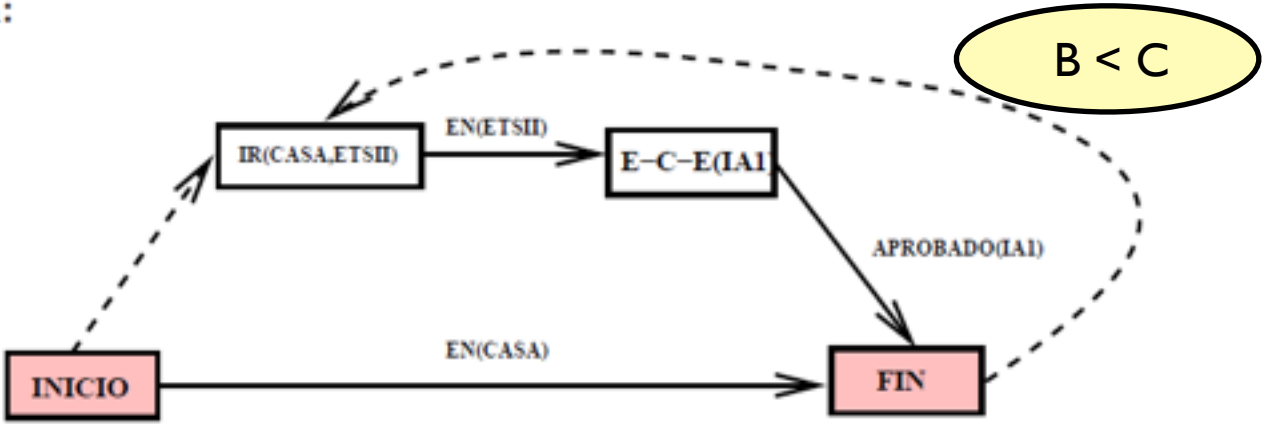


- Aparece una amenaza: de IR(CASA,ETSII) sobre el enlace causal entre INICIO y FIN
- Intentamos resolver la amenaza por promoción o por degradación

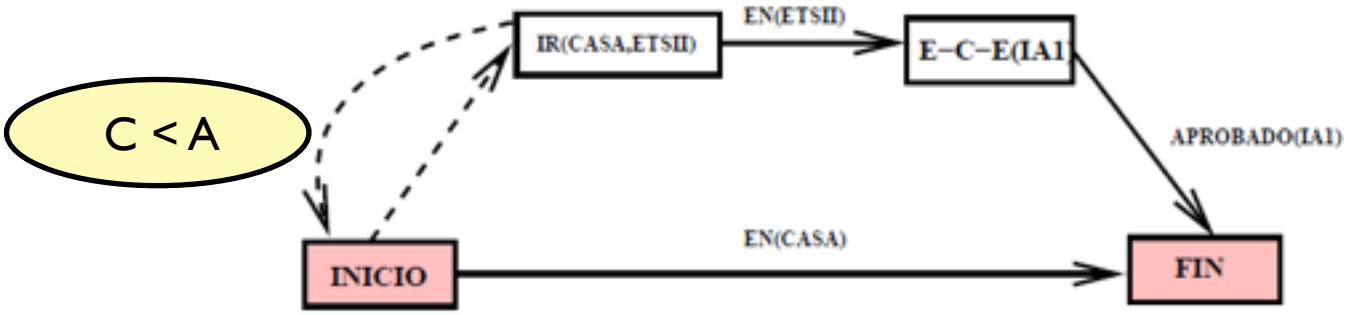
```
IR(x,y)          ESTUDIAR(x)          EXAMINARSE-CON-EXITO(x)
P: EN(x)           P: {}                P: EN(ETSII), ESTUDIADO(x)
B: EN(x)           B: {}                B: {}
A: EN(y)           A: ESTUDIADO(x)      A: APROBADO(x)
```

Ejemplo: aprobar IAIC

- Pero en ambos casos, se crearía un ciclo:
  - Promoción:



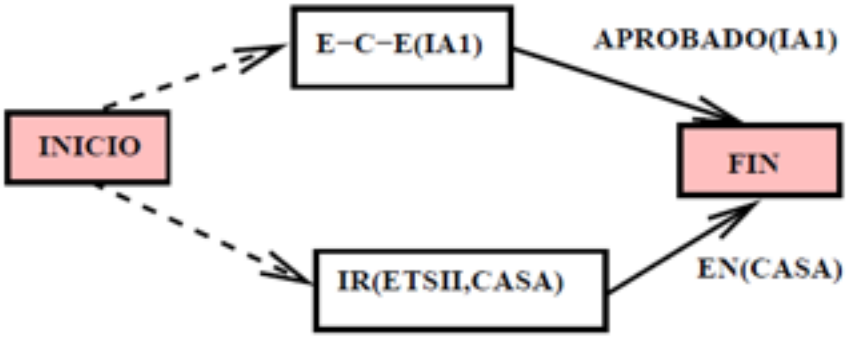
  - Degradación:




```
IR(x,y)          ESTUDIAR(x)          EXAMINARSE-CON-EXITO(x)
P: EN(x)           P: {}                P: EN(ETSII), ESTUDIADO(x)
B: EN(x)           B: {}                B: {}
A: EN(y)           A: ESTUDIADO(x)      A: APROBADO(x)
```

Ejemplo: aprobar IAIC

- Por tanto, hay FALLO y hemos de retroceder al anterior punto de ramificación, que estaba en el paso 2
  - Y elegir la otra alternativa: acción nueva IR(ETSII,CASA) para resolver la precondición `EN(CASA)` de FIN
- Paso 5:

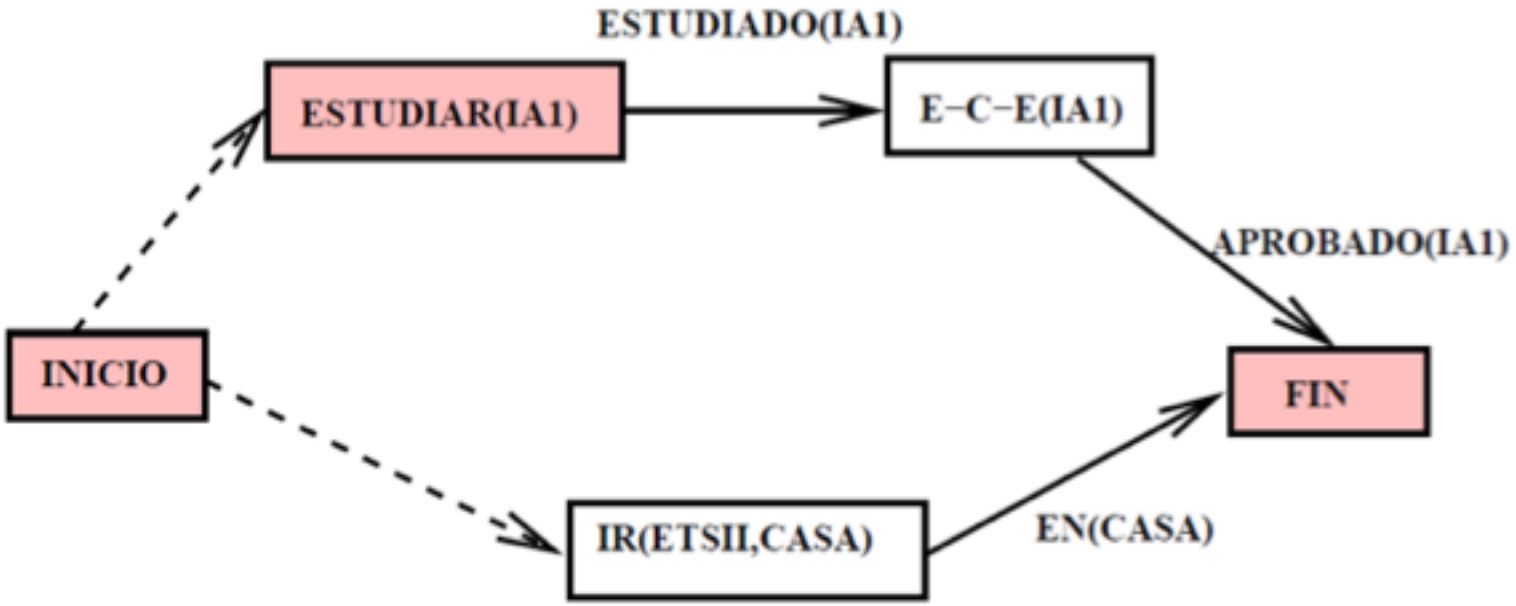


- Elegimos ahora la precondición `ESTUDIADO(IA1)` de EXAMINARSE-CON-EXITO(IA1)
  - La única posibilidad para resolverla es acción nueva ESTUDIAR(IA1)

```
IR(x,y)          ESTUDIAR(x)          EXAMINARSE-CON-EXITO(x)
P: EN(x)           P: {}                P: EN(ETSII), ESTUDIADO(x)
B: EN(x)           B: {}                B: {}
A: EN(y)           A: ESTUDIADO(x)      A: APROBADO(x)
```

Ejemplo: aprobar IAIC

- Paso 6:

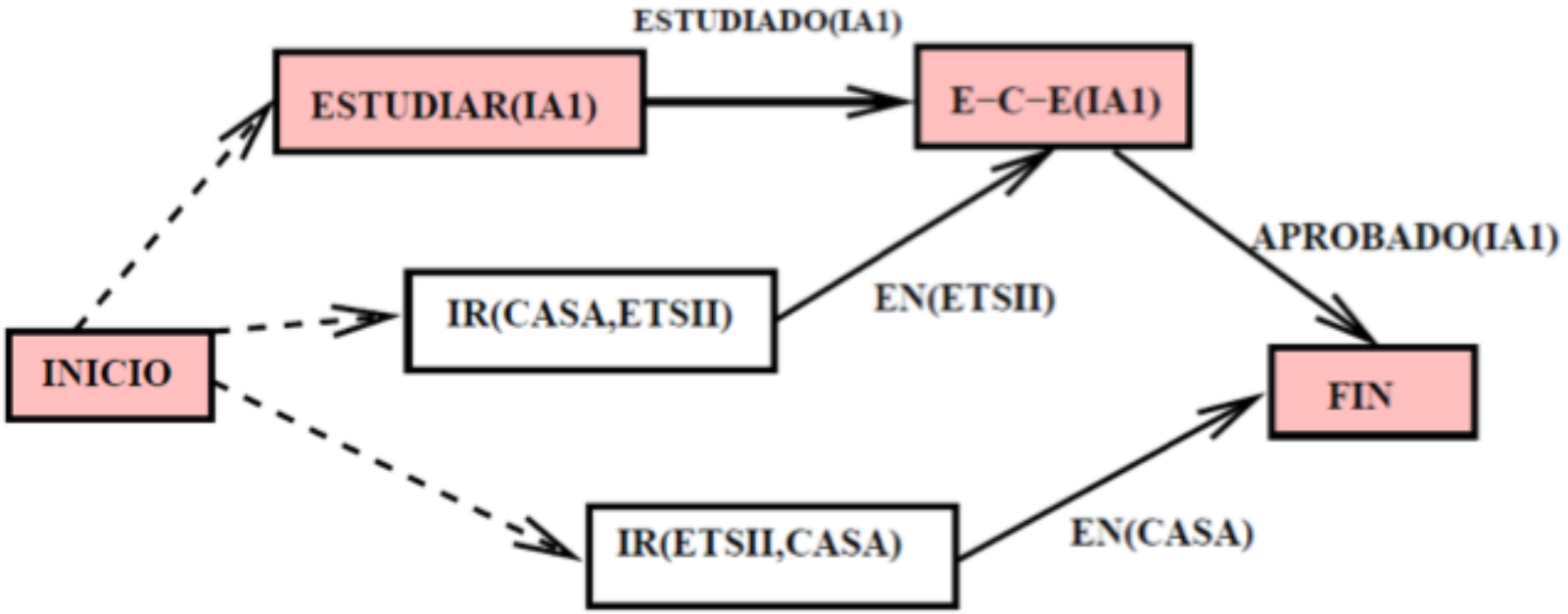


- Elegimos ahora la precondición EN(ETSII) de EXAMINARSE-CON-EXITO(IA1)
  - La única posibilidad para resolverla es acción nueva IR(CASA,ETSII)

```
IR(x,y)          ESTUDIAR(x)          EXAMINARSE-CON-EXITO(x)
P: EN(x)           P: {}                P: EN(ETSII), ESTUDIADO(x)
B: EN(x)           B: {}                B: {}
A: EN(y)           A: ESTUDIADO(x)      A: APROBADO(x)
```

Ejemplo: aprobar IAIC

- Paso 7:

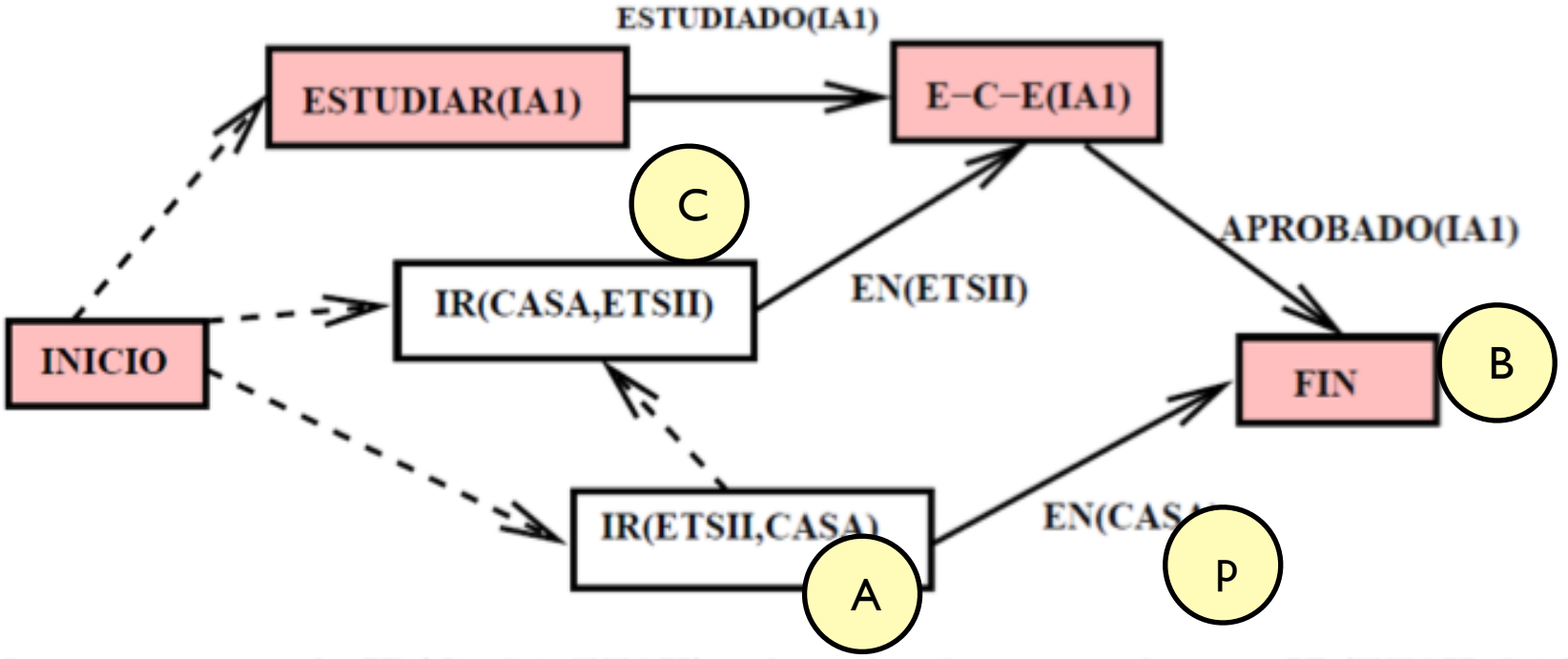


- Ahora existen amenazas : por ejemplo, de IR(ETSII,CASA) sobre el enlace causal entre IR(CASA,ETSII) y EXAMINARSE-CON-EXITO(IA1)
- Intentamos resolver esa amenaza por degradación; si no diera resultado, intentamos promoción

```
IR(x,y)          ESTUDIAR(x)          EXAMINARSE-CON-EXITO(x)
P: EN(x)           P: {}                P: EN(ETSII), ESTUDIADO(x)
B: EN(x)           B: {}                B: {}
A: EN(y)           A: ESTUDIADO(x)      A: APROBADO(x)
```

Ejemplo: aprobar IAIC

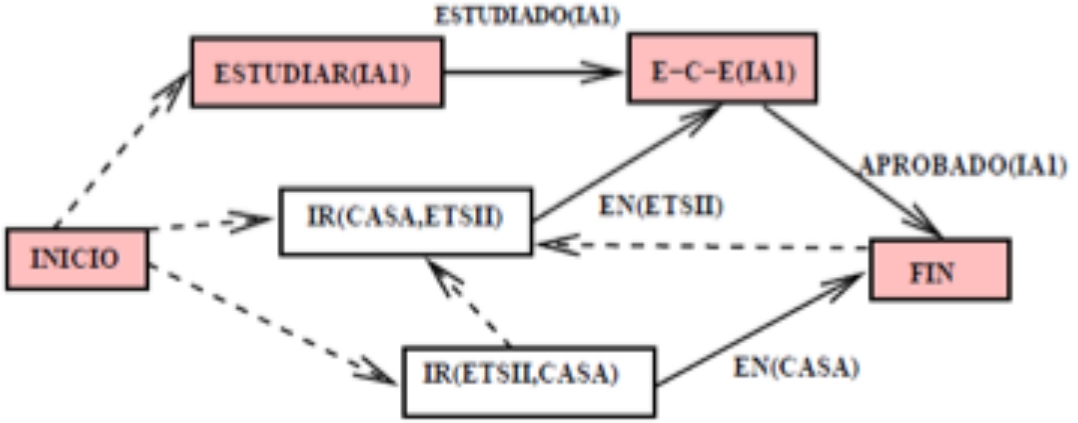
- Paso 8 (degradación):



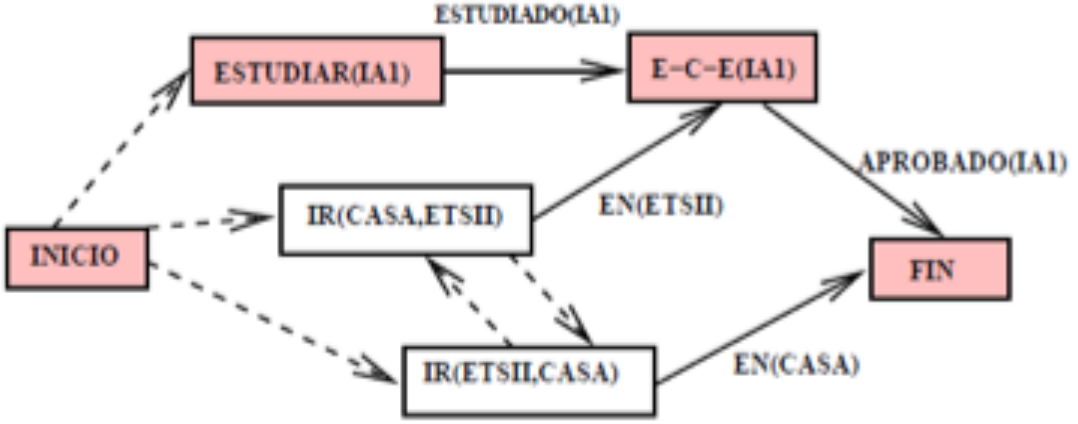
- Otra amenaza: de IR(CASA,ETSII) sobre el enlace causal entre IR(ETSII,CASA) y FIN
- Intentamos resolver la amenaza por promoción o por degradación

Ejemplo: aprobar IAIC

- Pero en ambos casos, se crearía un ciclo:
  - Promoción:

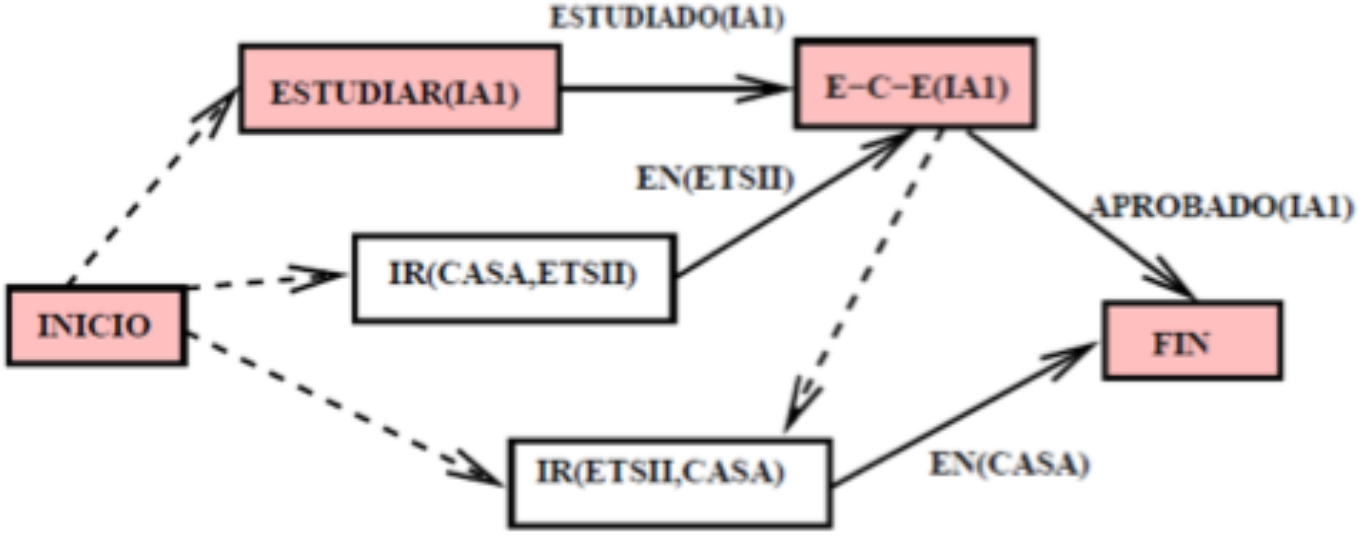


  - Degradación:



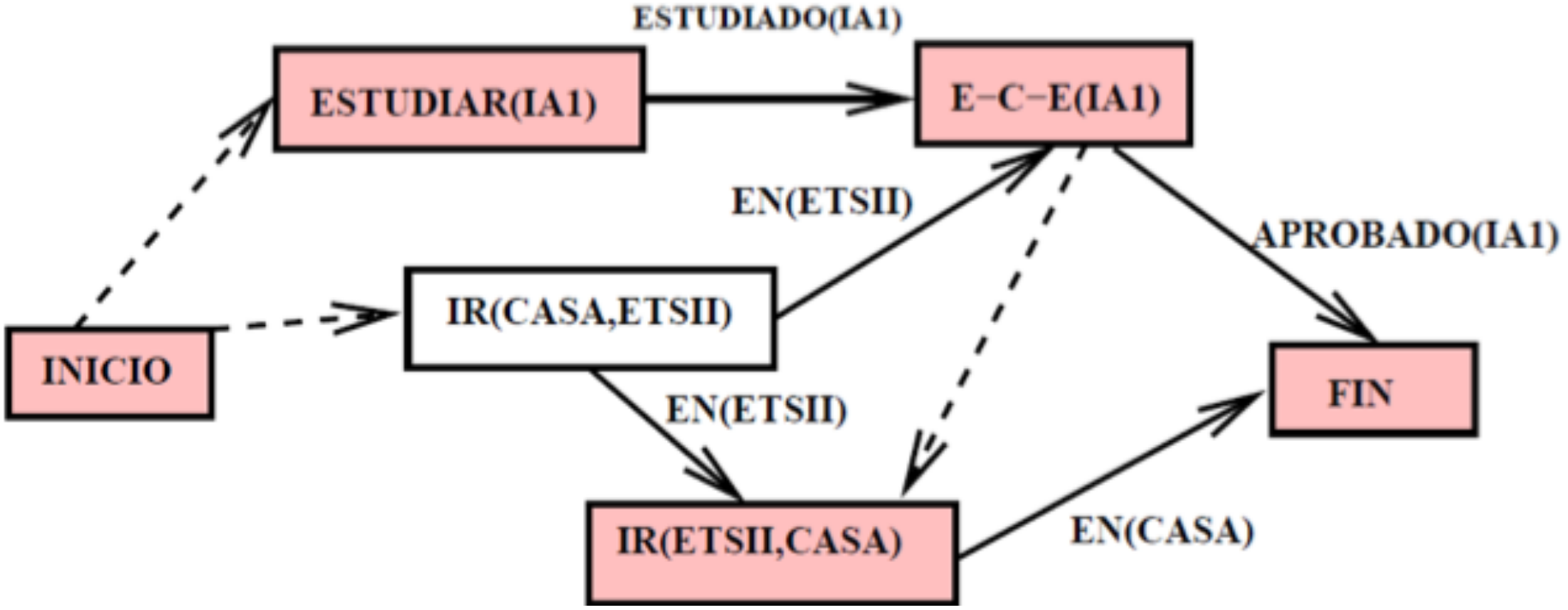
Ejemplo: aprobar IAIC

- Por tanto, hay FALLO
  - Hemos de retroceder al anterior punto de ramificación (paso 7), y elegir la otra alternativa: resolver la amenaza de IR(ETSII,CASA) sobre el enlace causal entre IR(CASA,ETSII) y EXAMINARSE-CON-EXITO(IA1), mediante promoción
- Paso 9:



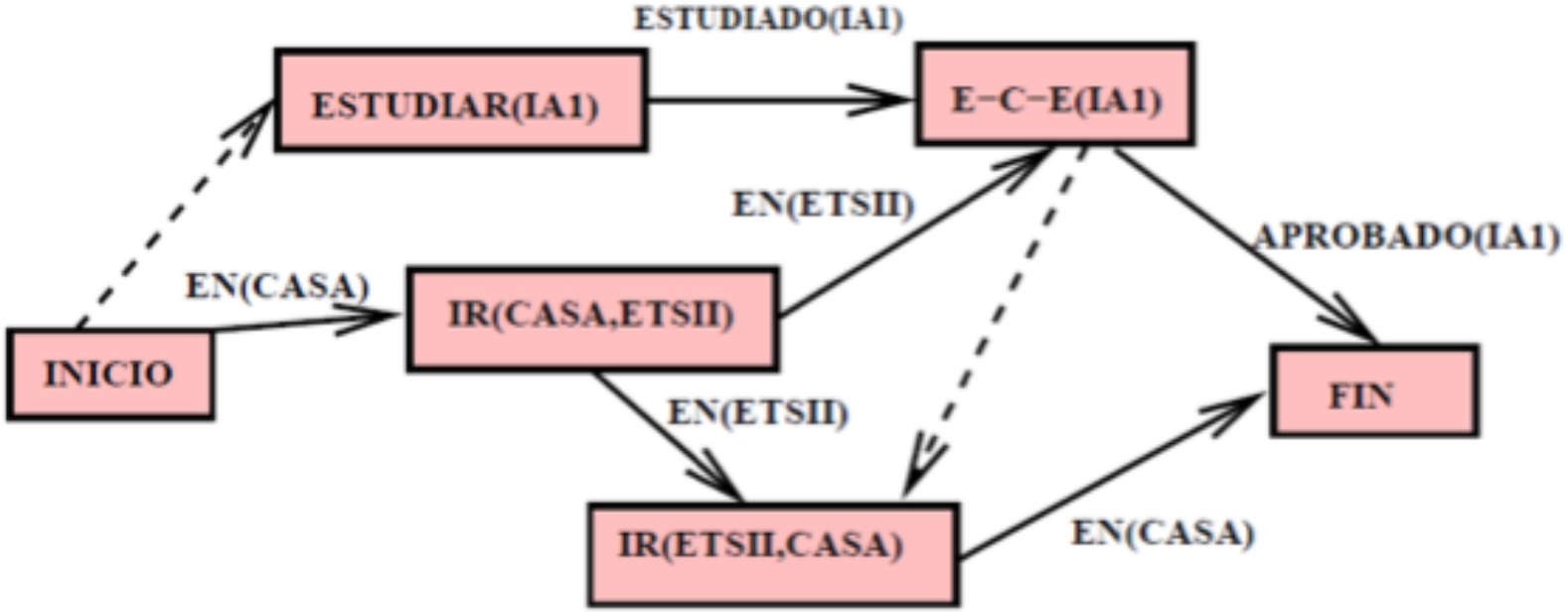
Ejemplo: aprobar IAIC

- Elegimos ahora la precondición EN(ETSII) de IR(ETSII,CASA)
  - Dos alternativas: establecimiento simple con IR(CASA,ETSII) o acción nueva con IR(CASA,ETSII).
  - Elegimos la primera, reconsiderándolo si fuera necesario
- Paso 10:



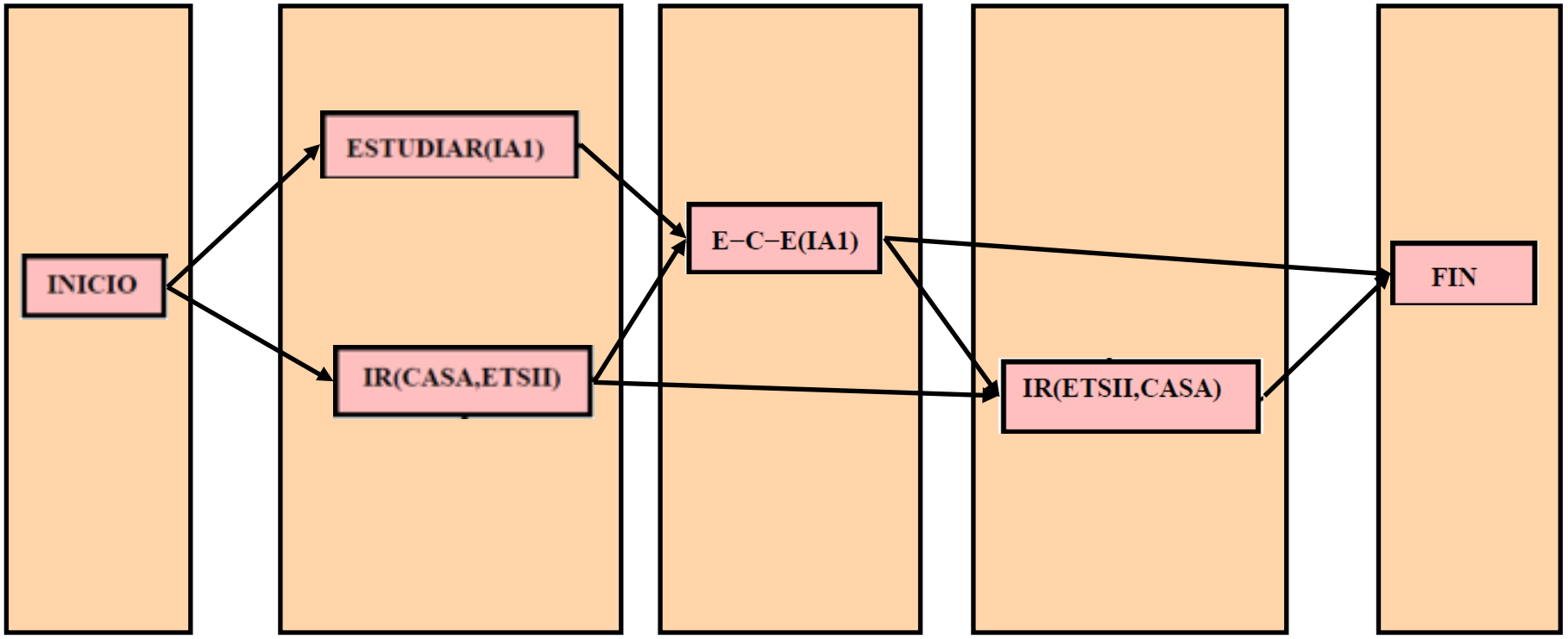
Ejemplo: aprobar IAIC

- Elegimos la precondición EN(CASA) de IR(CASA,ETSII)
  - Tres alternativas: establecimiento simple con INICIO, establecimiento simple con IR(ETSII,CASA) o acción nueva con IR(ETSII,CASA).
  - Elegimos la primera, reconsiderándolo si fuera necesario
- Paso 11 (Plan parcial final):



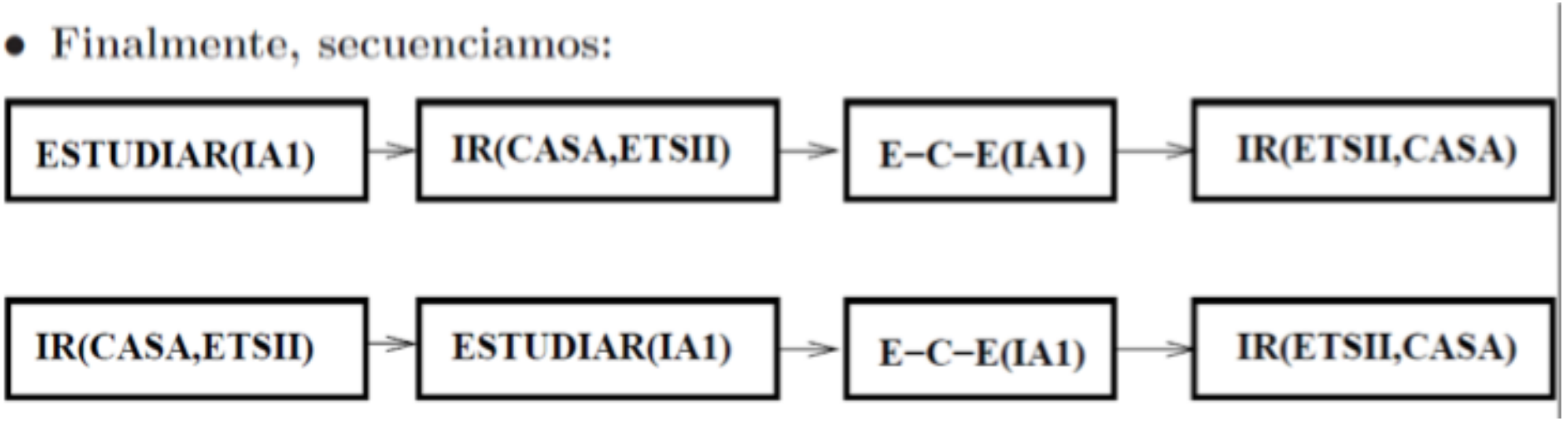
Secuenciación

- Pintar el grafo, con flechas hacia delante
- Escribir todos los nodos, respetando el orden entre “zonas” y aleatorio dentro de la misma.



Ejemplo: aprobar IAIC

- Finalmente, secuenciamos:



Algoritmo POP

FUNCION POP(COTA)

1. Hacer POP-REC(PLAN-INICIAL(*ESTADO-INICIAL*,*OBJETIVOS*),COTA)

FUNCION POP-REC(PLAN,COTA)

1. Si PLAN es final, devolver SECUENCIACION(PLAN) y terminar.
2. Si existe en PLAN una amenaza de la acción C sobre A --p-->B, entonces
   - 2.1 Sean PLANPR obtenido de PLAN mediante promoción en la amenaza anterior
   - 2.2 Si PLANPR tiene ciclos, sea RESULTADO igual a FALLO; en caso contrario, sea RESULTADO igual a POP-REC(PLANPR,COTA).
   - 2.3 Si RESULTADO no es FALLO, devolver RESULTADO y terminar; en otro caso:
     - 2.3.1 Sea PLANDEG obtenido de PLAN mediante degradación en la amenaza anterior
     - 2.3.2 Si PLANDEG tiene ciclos, sea RESULTADO igual a FALLO; en caso contrario, sea RESULTADO igual a POP-REC(PLANDEG,COTA).
     - 2.3.3 Devolver RESULTADO y terminar.
3. Sea P una precondición abierta en PLAN
4. Para cada acción (nueva o ya existente) que tiene a P como efecto, hacer:
   - 4.1 Hacer PLANEXT el resultado de cerrar la precondición P en PLAN mediante un enlace causal (si fueran acción nueva, añadir también la acción al PLAN)
   - 4.2 Si PLANEXT no tiene ciclos y un número de acciones menor o igual que COTA, entonces:
     - 4.2.1 Sea RESULTADO igual POP-REC(PLANEXT,COTA)
     - 4.2.2 Si RESULTADO no es FALLO, devolver RESULTADO y terminar
5. Devolver FALLO (* aquí sólo se llegaría si la precondición abierta no se puede resolver con éxito mediante ninguna acción *)

the form of networks.

Planning problems are specified in the hierarchical task network approach by providing a set of tasks, which can be:

1. primitive tasks, which roughly correspond to the actions of STRIPS;
2. compound tasks, which can be seen as composed of a set of simpler tasks;
3. goal tasks, which roughly corresponds to the goals of STRIPS, but are more general.

A primitive task is an action that can be executed. A compound task is a complex task composed of a sequence of actions. A goal task is a task of satisfying condition. The difference between primitive and other tasks is that the primitive actions can be directly executed. Compound and goal tasks both require a s of primitive actions to be performed; however, goal tasks are specified in terms of conditions that have to be made true, while compound tasks can only be specified in terms of other tasks via the task network outlined below.

Constraints among tasks are expressed in form of networks, called task networks. A task network is a set of tasks and constraints among them. Such a netw be used as the precondition for another compound or goal task to be feasible. This way, one can express that a given task is feasible only if a set of other ac (those mentioned in the network) are done, and they are done in such a way that the constraints among them (specified by the network) are satisfied. One pa formalism for representing hierarchical task networks that has been fairly widely used is TAEMS.

A task network can for example specify that a condition is necessary for a primitive action to be executed. When this network is used as the precondition fo compound or goal task, it means that the compound or goal task requires the primitive action to be executed and that the condition must be true for its exec successfully achieve the compound or goal task.

The best-known domain-independent HTN-planning software is:

- Nonlin, one of the first HTN planning systems.[1]
- SIPE-2[2]
- O-Plan[3]
- UMCP, the first provably sound and complete HTN planning systems.[4]
- SHOP2, a HTN-planner developed at University of Maryland, College Park.[5]
- HTNPlan-P, preference-based HTN planning.[6]

HTN is a useful way to provide the planning engine with information about the hierarchical structure of the planning domain. HTN-like planning (as it is practically used) has the same expressivity (i.e. can solve the same domains) as STRIPS.[7] The theoretical model of HTN is strictly more expressive than S [8] but cannot be directly used because of its undecidability.

See also

- STRIPS
- Hierarchical control system - a feedback control system well suited for HTN planning

References

1. **Jump up ^** Nonlin
2. **Jump up ^** David E. Wilkins. "SIPE-2: System for Interactive Planning and Execution". *Artificial Intelligence Center*. SRI International. Retrieved 2013-06-13.

- Simple Hierarchical Ordered Planner

- SHOP (Simple Hierarchical Ordered Planner), JSHOP, and SHOP2 are domain-independent automated-planning systems. They are based on *ordered task decomposition*, which is a type of *Hierarchical Task Network (HTN)* planning.

In **HTN planning**, the planning system begins with an initial state-of-the-world and with the objective of creating a plan to perform a set of *tasks* (abstract representations of things that need to be done). HTN planning is done by problem reduction: the planner recursively decomposes tasks into subtasks, stopping when it reaches *primitive* tasks that can be performed directly by *planning operators*. In order to tell the planner how to decompose nonprimitive tasks into subtasks, it needs to have a set of *methods*, where each method is a schema for decomposing a particular kind of task into a set of subtasks (provided that some set of preconditions is satisfied). For each task, there may be more than one applicable method, and thus more than one way to decompose the task into subtasks. The planner may have to try several of these alternative decompositions before finding one that is solvable at a lower level. Unlike classical planning, HTN planning is Turing-complete.

An **ordered task decomposition** planner is an HTN planner that plans for tasks in the same order that they will be executed. This reduces the complexity of reasoning by removing a great deal of uncertainty about the world, which makes it easy to incorporate substantial expressive power into the planning algorithm. In addition to the usual HTN methods and operators, our planners can make use of axioms, can do mixed symbolic/numeric conditions, and can do external function calls.

Our first planning systems to use ordered task decomposition were *domain-specific*, i.e., they were tailor-made for specific application domains. These included EDAPS, a system for integrated product design and manufacturing planning, and Bridge Baron, a bridge-playing computer program that won the 1997 world championship of computer bridge and is a successful commercial product.

Our first *domain-independent* implementation of ordered task decomposition was SHOP (Simple Hierarchical Ordered Planner). SHOP and its successors (the most recent ones are SHOP2 and JSHOP2) can be configured to work in many different planning domains.

In the 2002 International Planning Competition, SHOP2 solved problems in every planning domain in the competition, for a total of nearly 1000 planning problems. SHOP2 received one of the top four prizes in the competition.

Our most recent planner is JSHOP2, a Java implementation of SHOP2 (which is written in Lisp). In addition to being in Java, JSHOP2

- **O-Plan - Open Planning Architecture**

Outline of Work on Multi-Perspective Planning
<I-N-OVA> Constraint Model of Activity and <I-N-CA>
Technical Approach
Applications of O-Plan
Acknowledgements
Contact

System Availability

O-Plan - Open Planning Architecture

O-Plan: Current Release Information
O-Plan: Demonstrations
O-Plan Introductory Demonstration Page - AAAI-2000 Intelligent Systems Demonstration of O-Plan
O-Plan: Documents Available
Common Process Framework - Steve Polyak
Capability Description Language - Gerhard Wickler

Pacifica - Evaluation and Demonstration Domain

The Pacifica Environment
Pacifica Simulator and WorldSim - a general-purpose simulation environment - Glen Reece

DARPA/Air Force Research Laboratory Planning Initiative (ARPI)

AIAI Role in ARPI
Tate, A. (ed.), Advanced Planning Technology, AAAI Press, May 1996.

Intelligent Planning. Qiang Yang. Springer Verlang, Germany, 1977.
Generating Abstraction Hierarchies. Craig A Knoblock. Kluwer Academic Publishers, EEUU, 1993.
New Directions in AI Planning. Edited by M. Ghallab and A. Milani. Publisher IOS Press. Amsterdem, 1996.
K. Erol. Hierarchical Task Network Planning: Formalization, Analysis, and Implementation. PhD thesis, University of Maryland, 1996.
Inteligencia Artificial, un Enfoque Moderno; Russel Stuart y Peter Norving; Ed. Prentice Hall-1996.
Inteligencia Artificial; Elaine Rich y Kevin Knisht; ed. McGraw Hill-1997.
Inteligencia Artificial, una nueva síntesis; Nils J. Nilsson; Ed. McGraw Hill-2001
. Software Agents, J. M. Bradshaw, the MIT Press, 1997.
Case Based Reasoning, David Leake, The MIT Press.
Journal, Journal of AI Research, AAAI and IJCAI Proceedings
Documentos Entregados para su análisis en clase.
Direcciones en Internet: A continuación se hace referencias a algunas páginas en donde se encuentra bibliografía sobre diferentes tópicos de planificación en inteligencia artificial.

subir

Planificación Práctica

Planificadores prácticos

Estos planificadores funcionan en dominios complejos y reales y permite detectar las fallas de POP y sugerir las posibles ampliaciones necesarias.

Optimum-AIV es un planificador utilizado por la Agencia Espacial Europea como auxiliar en el armado, integración y verificación (AIV) de naves espaciales. Este sistema sirve tanto para elaborar planes como para monitorear su ejecución. El objetivo principal de Optimum-AIV es la rápida capacidad de reprogramación. Básicamente estas herramientas toman un plan de orden parcial completo y con él proceden a elaborar un programa que sea óptimo para dicho plan. Las acciones se consideran como objetos que consumen tiempo y tienen restricciones de ordenamiento; se hace caso omiso de los efectos que éstos producen.

Para que los sistemas de IA puedan desempeñarse satisfactoriamente en el mundo real es necesario su integración al ambiente en el cual funcionan. Es vital que el planificador tenga acceso a las bases de datos donde está toda la información sobre el proyecto. El lenguaje Strips resulta limitado para un dominio AIV debido a que no tiene capacidad para la expresión de los cuatro conceptos básicos siguientes:

Planes jerárquicos: una forma de manejar la creciente complejidad consiste en especificar planes según un distinto grado de detalle. Entre este nivel y el de las acciones que se ejecutan habrá una docena de niveles intermedios.
Condiciones complejas: los operadores Strips básicamente son de tipo propositivo. Aceptan variables, pero se utilizan de manera limitada (no existe cuantificación universal) y los operadores son incondicionales.
Tiempo: está basado en el cálculo de situaciones, da por sentado que todas las acciones se producen instantáneamente y que una acción sigue a la otra, sin que haya lapsos entre ambas. Para los proyectos del mundo real es necesario contar con un mejor modelo para el tiempo donde sea capaz de representar que los proyectos tienen un límite de tiempo, de que las acciones tienen una duración y de que hay condiciones temporales para los lapsos de un plan.
Recursos: las limitaciones en cuanto a recursos se pueden expresar mediante la cantidad de objetos que puedan ser utilizados en un momento dado (por ejemplo personas) o una cantidad total de la que se puede disponer (por ejemplo dinero).

Optimum-AIV se basa en la arquitectura de planificación abierta O-Plan, este es semejante al planificador POP sólo que tiene capacidad para trabajar con un lenguaje más expresivo que puede representar tiempo, recursos y planes jerárquicos; también puede trabajar con heurísticas para guiar la búsqueda y lleva un registro de los motivos de cada elección, todo lo cual facilita la replanificación cuando sea necesario.

Programación de actividades: se puede dividir en una actividad de planificación (definir qué pasos de armado son los que van a llevarse a cabo) y una actividad de programación (definir cuándo y dónde se ejecutarán cada uno de tales pasos). En muchas fábricas modernas, la planificación se realiza a mano y para la programación se utiliza herramientas automatizadas. O-Plan es usado por Hitachi en la planificación y en la programación de actividades, mediante un sistema denominado TOSCA.

SIPE es un sistema para la planificación interactiva y el monitoreo de la ejecución, fue el primero en abordar el problema de la replanificación y el primero en emprender acciones en relación con el problema de los operadores expresivos. Es similar a O-Plan en cuanto a rango de sus