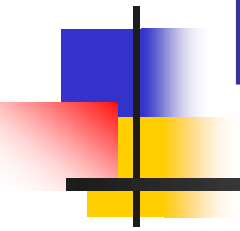
Programación Automática

MÁSTER EN CIENCIA Y TECNOLOGÍA INFORMÁTICA

Ricardo Aler Mur

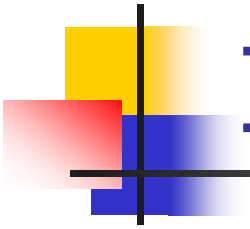


Programación Automática

Ricardo Aler Mur

Universidad Carlos III de Madrid

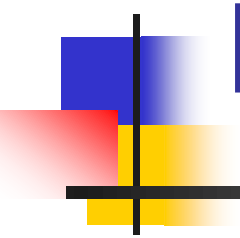
http://www.uc3m.es/uc3m/dpto/INF/aler

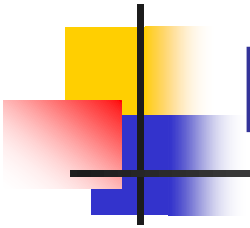


Índice

1. Learning by Example, learning by demonstration

Learning by Example, learning by demonstration

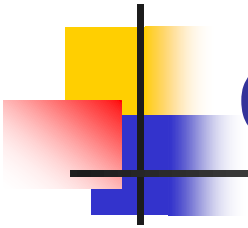




Bibliografía

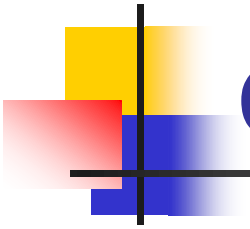
- Watch What I do. Programming by Demonstration [Cypher, 93]
- Your Wish is My Command. Programming by Example [Lieberman, 01]

Programming by demonstration (by example)

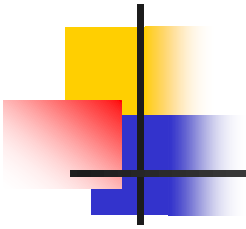


- Intenta programar siguiendo la relación maestro-alumno, sin necesidad de usar lenguajes informáticos
- El usuario muestra a la máquina como resolver el problema con uno o varios **ejemplos**
- La **traza** (pasos) de la ejecución está disponible

Programming by demonstration (by example)

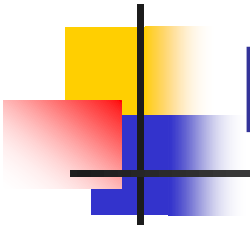


- Ell sistema intenta **generalizar** y construir el procedimiento genérico (con repeticiones y control de flujo)
- El sistema puede ejecutar lo aprendido, para que el maestro **le corrija** ante los errores
- **Gráfico**, **interactivo**



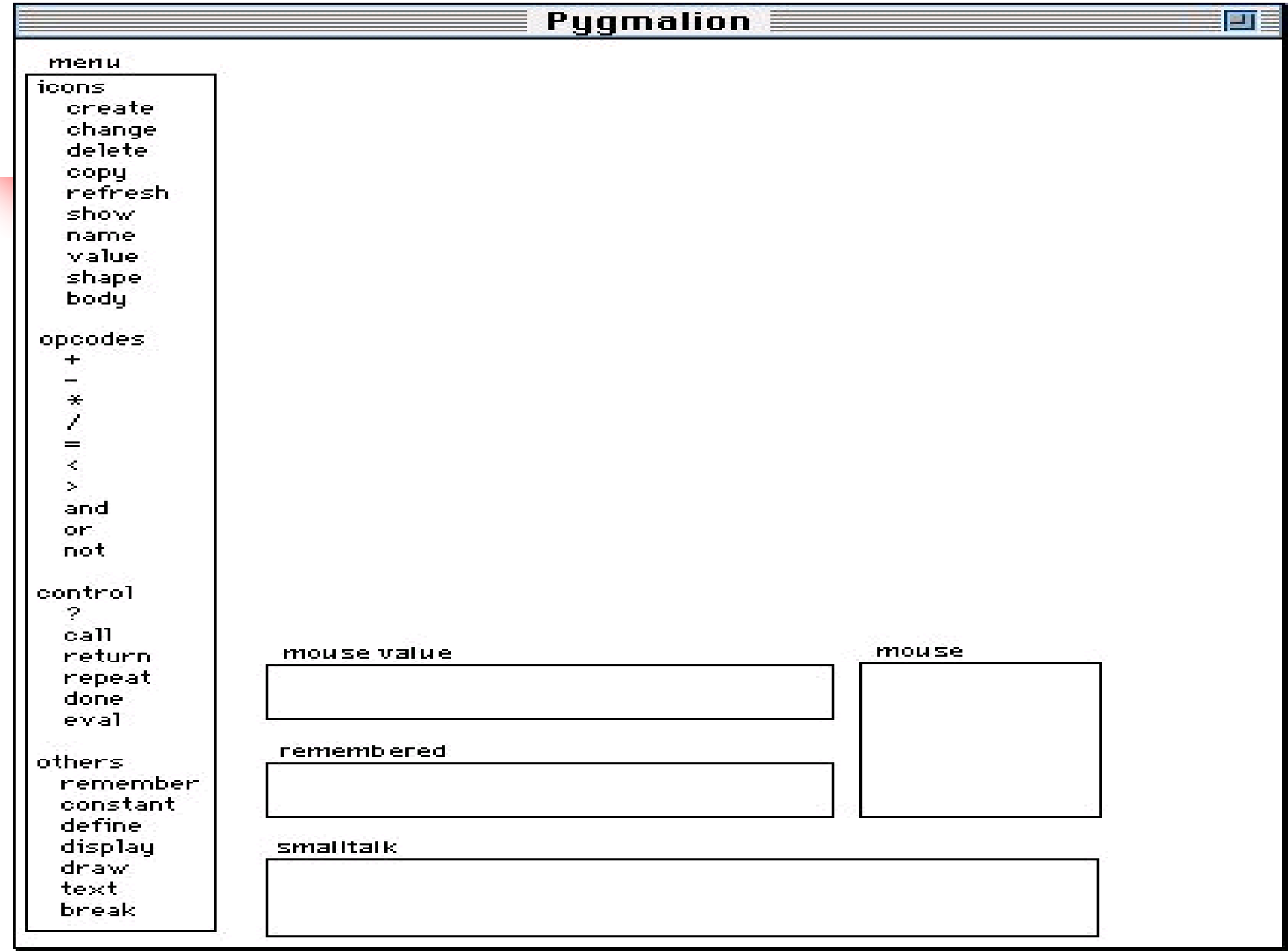
PBD. Ejemplos

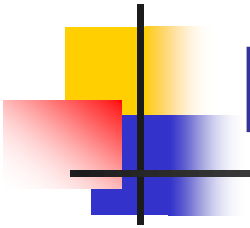
- Pygmalion [Cainfield, 1975]:
  - Se le muestra a la máquina cómo programar el factorial de manera gráfica, **con un caso concreto** (6!)
  - Comunicación a través de una pizarra
  - Objetivo: que no-programadores puedan programar de manera gráfica
  - Ejecuta programas parcialmente definidos y descubre trozos de código faltantes, y le pide al usuario que los rellene



Pygmalion. Factorial

- Enseñar a Pygmalion a programar el factorial mediante un caso concreto (6!)
- Factorial (n) =
  - If (n=0) then 1
  - Else n*factorial(n-1)


Pygmalion
menu
icons
create
change
delete
copy
refresh
show
name
value
shape
body
opcodes
+
-
*
/
=
<
>
and
or
not
control
?
call
return
repeat
done
eval
others
remember
constant
define
display
draw
text
break
mouse value
mouse
remembered
smalltalk



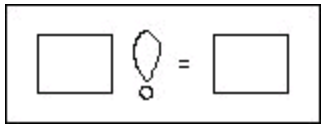
La pizarra de Pygmalion

- Menu: iconos y operaciones sobre iconos. Los iconos pueden tener código y datos asociados
- Mouse value: si se teclea un valor ahí, se queda pegado al ratón
- Remembered: se acuerda de las dos últimas operaciones

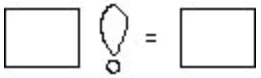
Pasos para definir el factorial

1. Se define un icono para la función

2. Se crean dos subiconos para representar el argumento, la operación y la solución



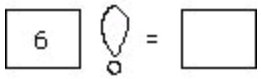
3. Se quita el borde externo



Pasos para definir el factorial

4. Mediante define, se asocia el icono con el nombre de la función (factorial)

5. Se escribe el valor con el que se va a ilustrar la función factorial

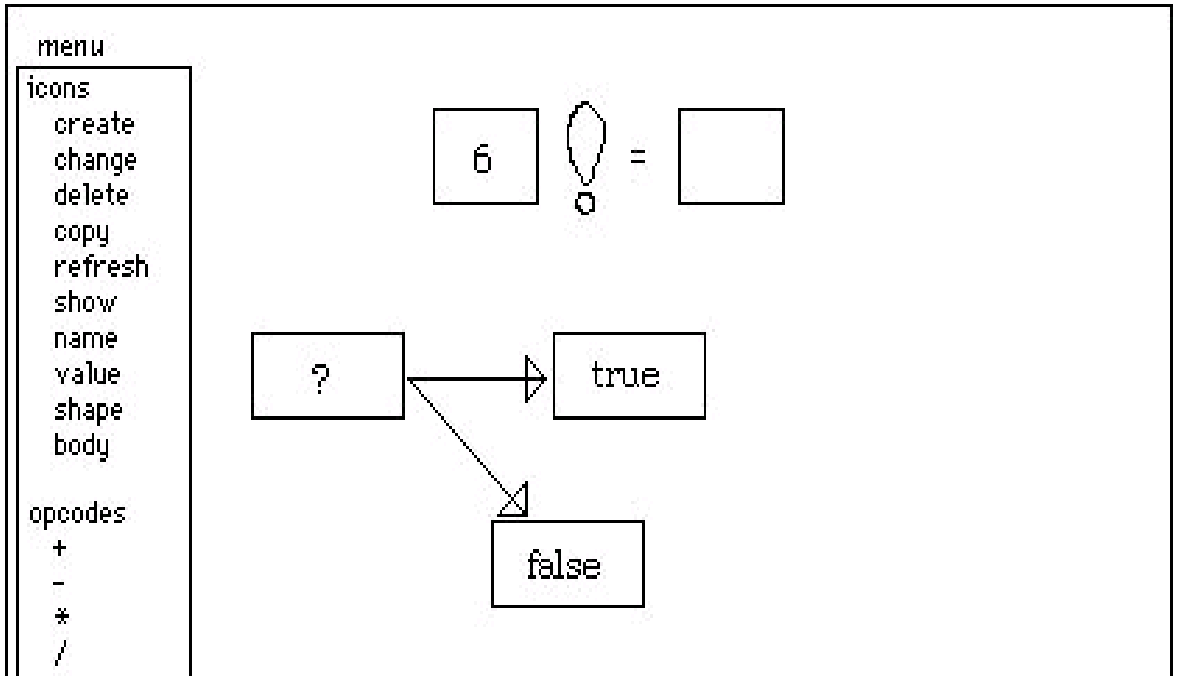


6. Como el argumento está relleno, el factorial se ejecuta

7. Pero como el cuerpo está vacío, le pregunta al usuario qué hacer

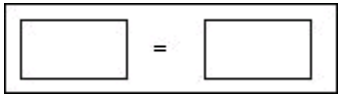
Pasos para definir el factorial

8. El programador decide que hace falta un condicional

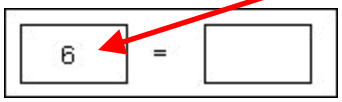


Pasos para definir el factorial

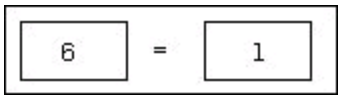
9. Se rellena la condición

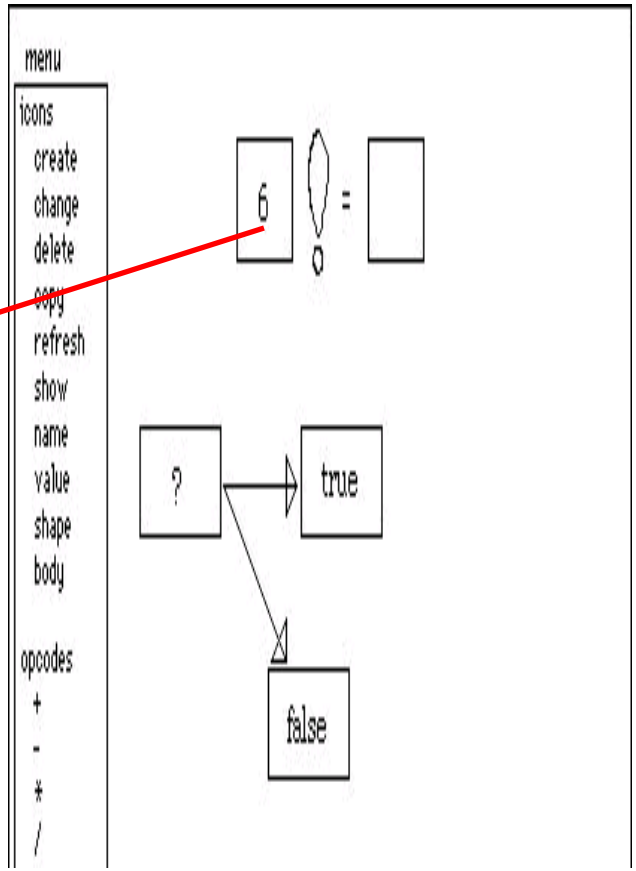


10. Se arrastra el 6 hasta la condición



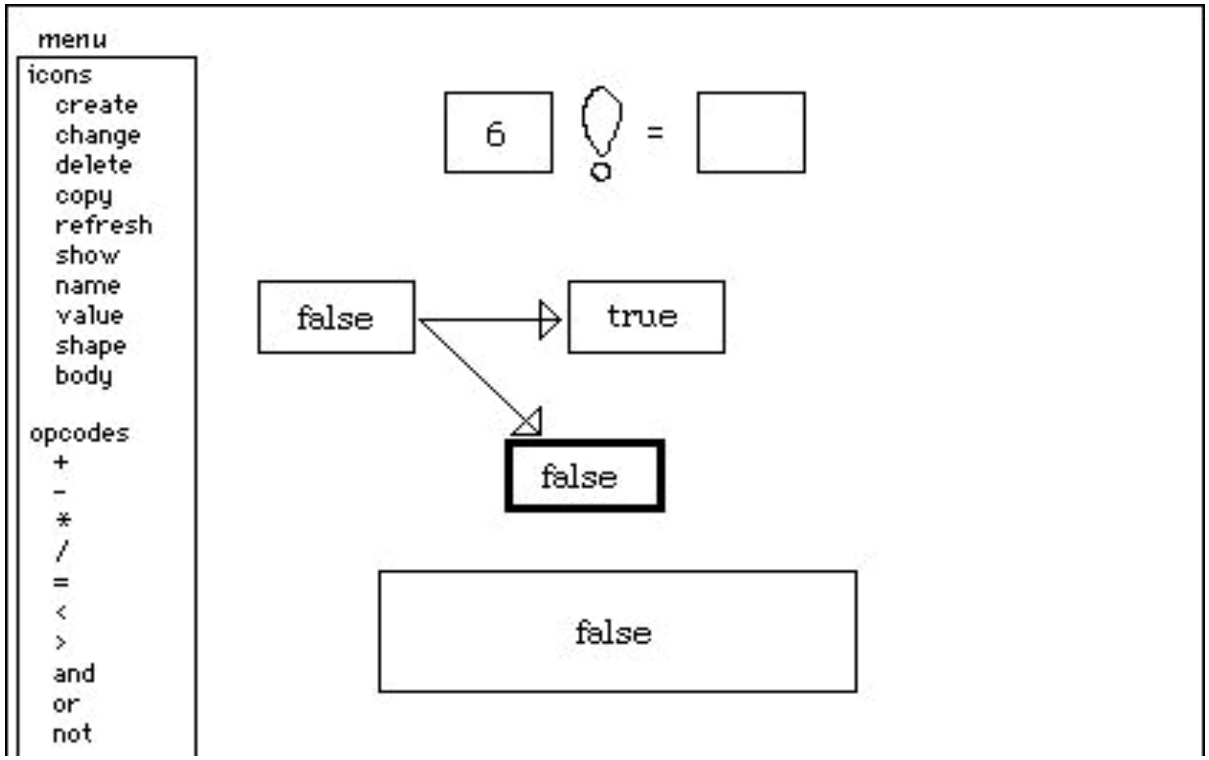
11. Se compara con 1





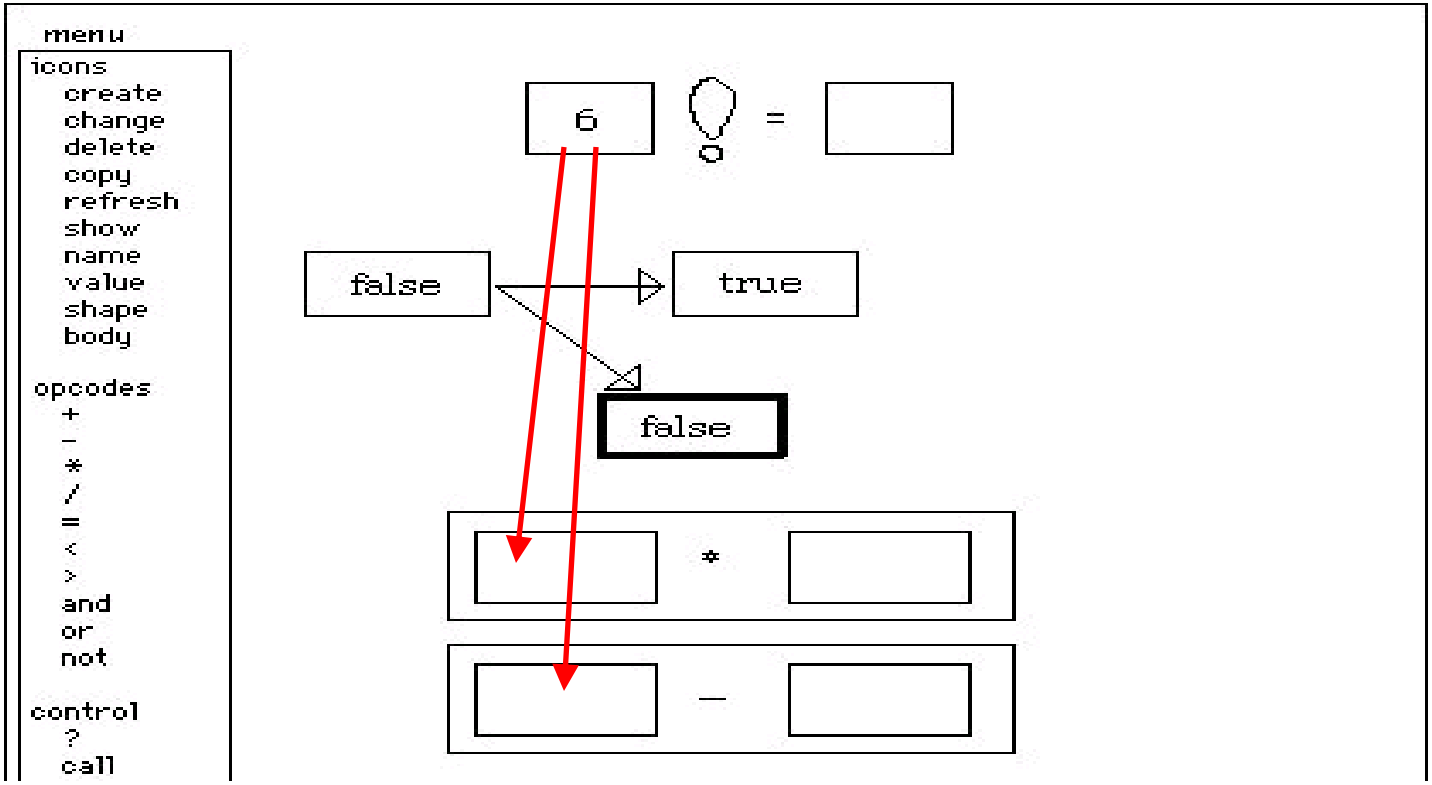
Pasos para definir el factorial

12. Se ejecuta la condición, pero el cuadrado de **false** no tiene código, luego pregunta al usuario



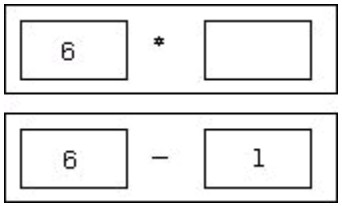
Pasos para definir el factorial

13. Se define código para el **false** (n*factorial(n-1))

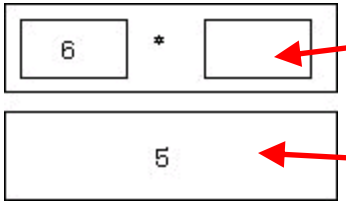


Pasos para definir el factorial

14. Se rellenan los argumentos



15. Se ejecuta automáticamente la resta

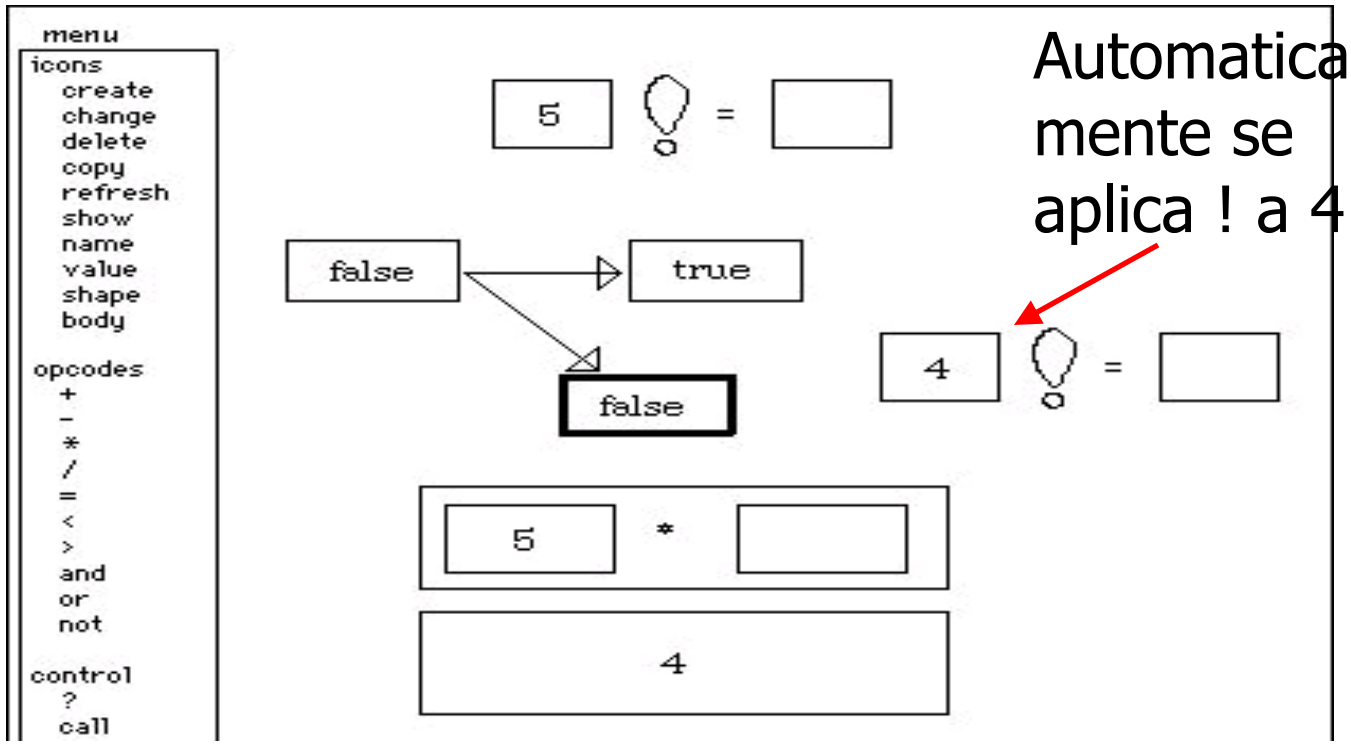


Mas adelante, pondremos aquí el resultado de 5!

Aplicamos factorial aquí

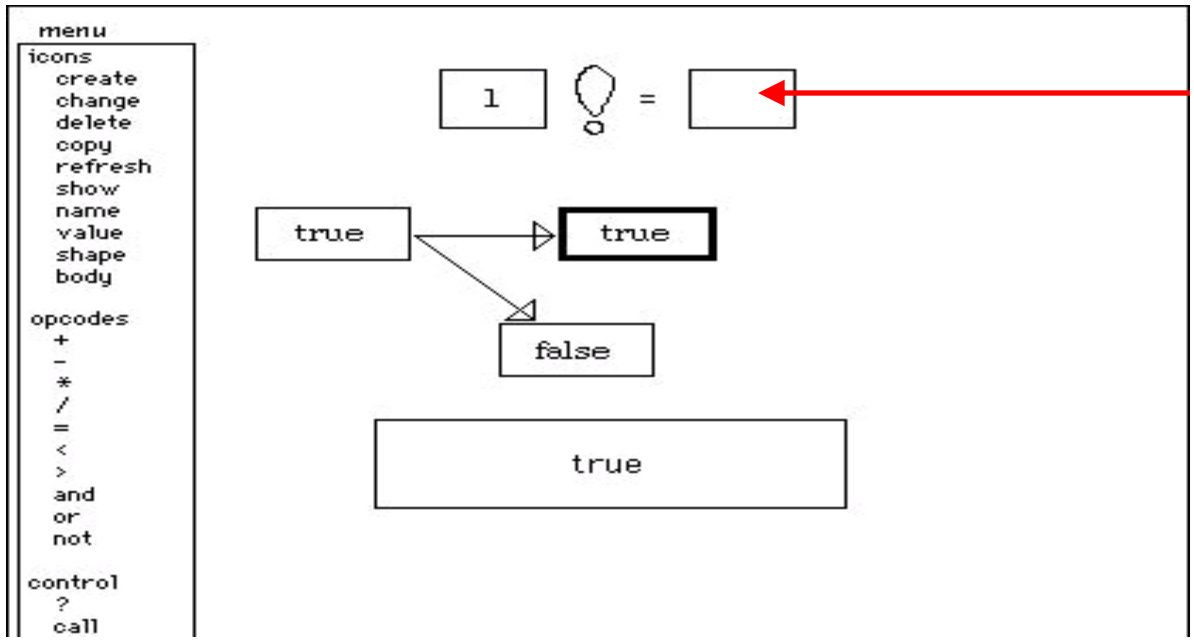
Pasos para definir el factorial

16. Se aplica factorial a 5 y se ejecuta automáticamente

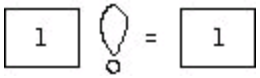


Pasos para definir el factorial

17. Y así sucesivamente, hasta llegar a 1

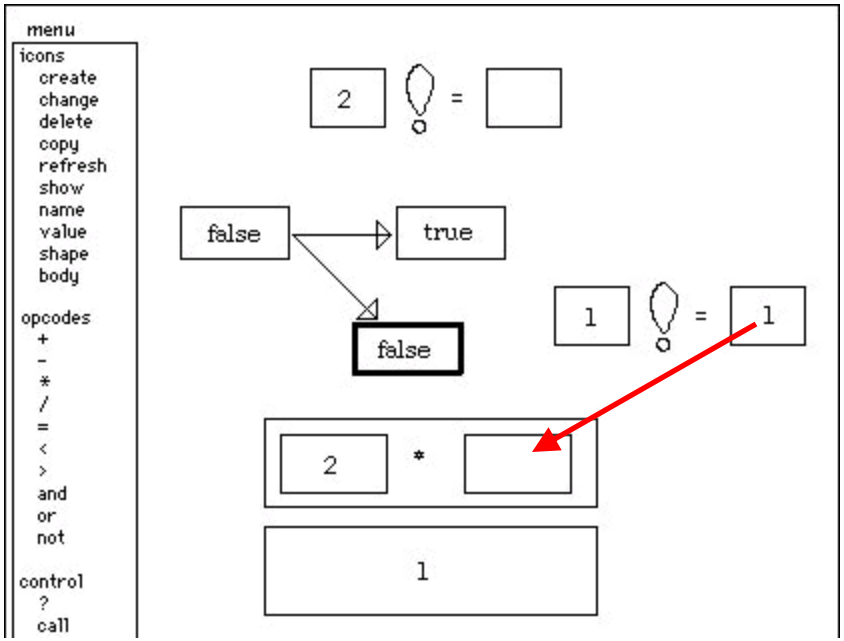


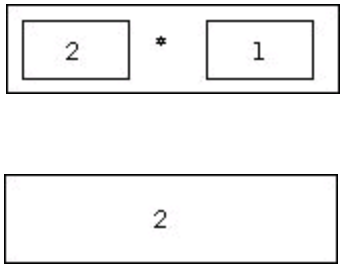
Ponemos 1 ahí:



Pasos para definir el factorial

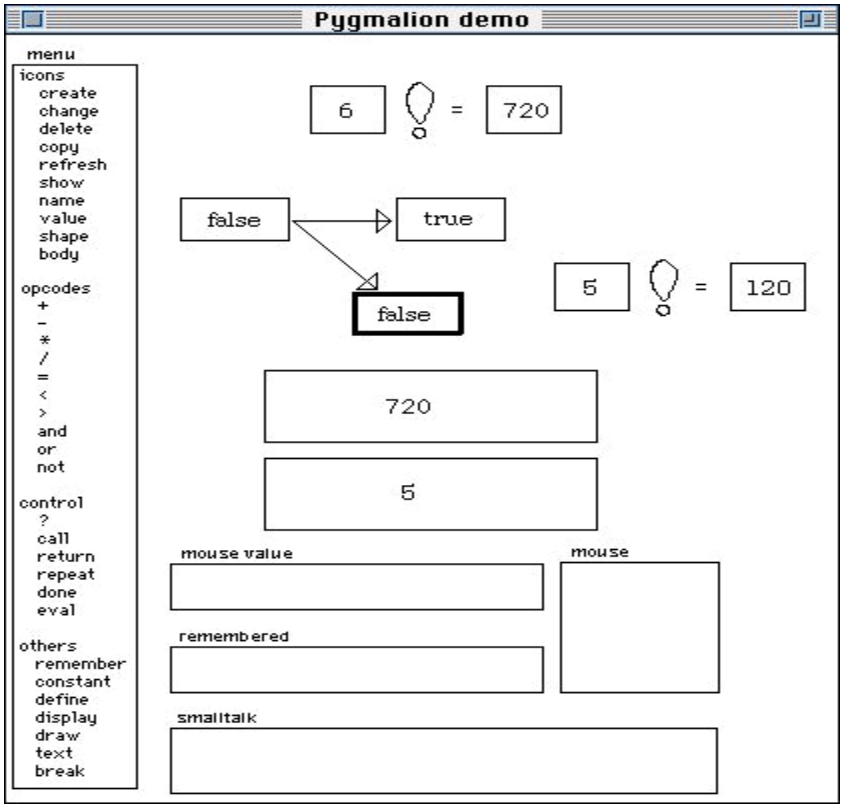
18. Esto nos devuelve al factorial de 2. Arrastramos el 1 al hueco

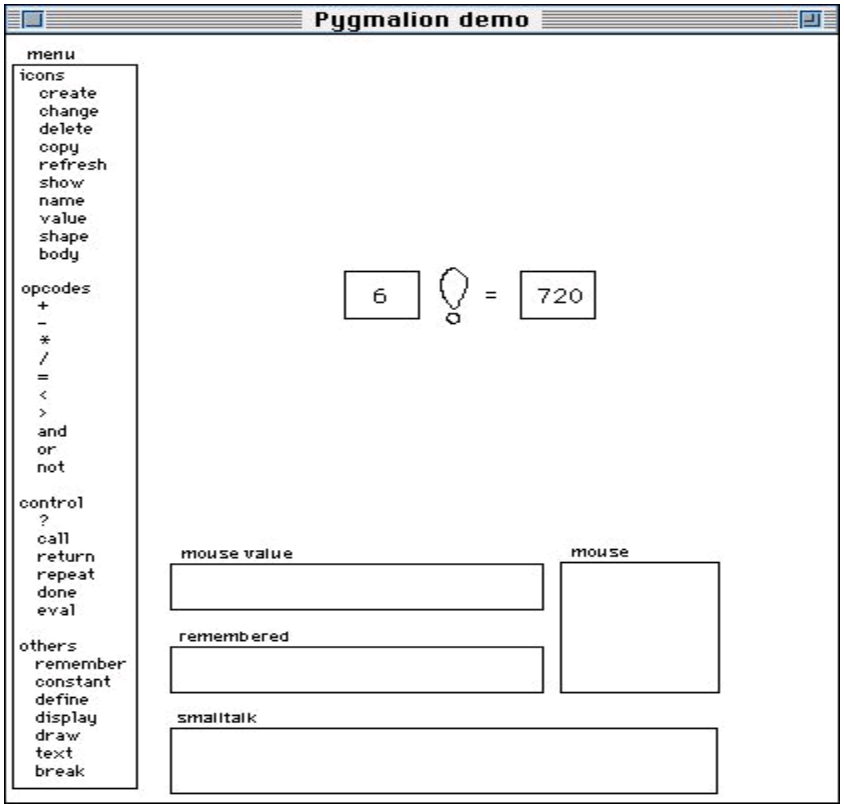


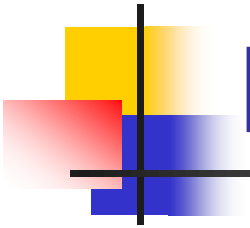


Pasos para definir el factorial

19. Ahora ya sabe hacer todo y calcula el factorial. Además, se tiene la **definición** del factorial



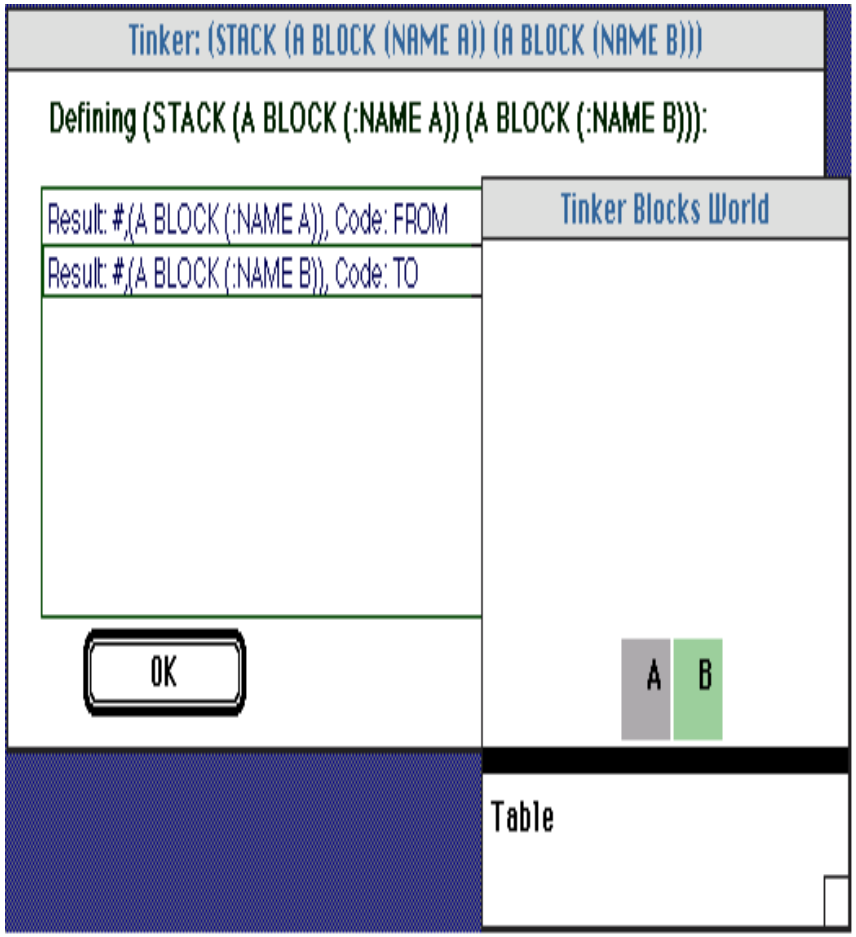


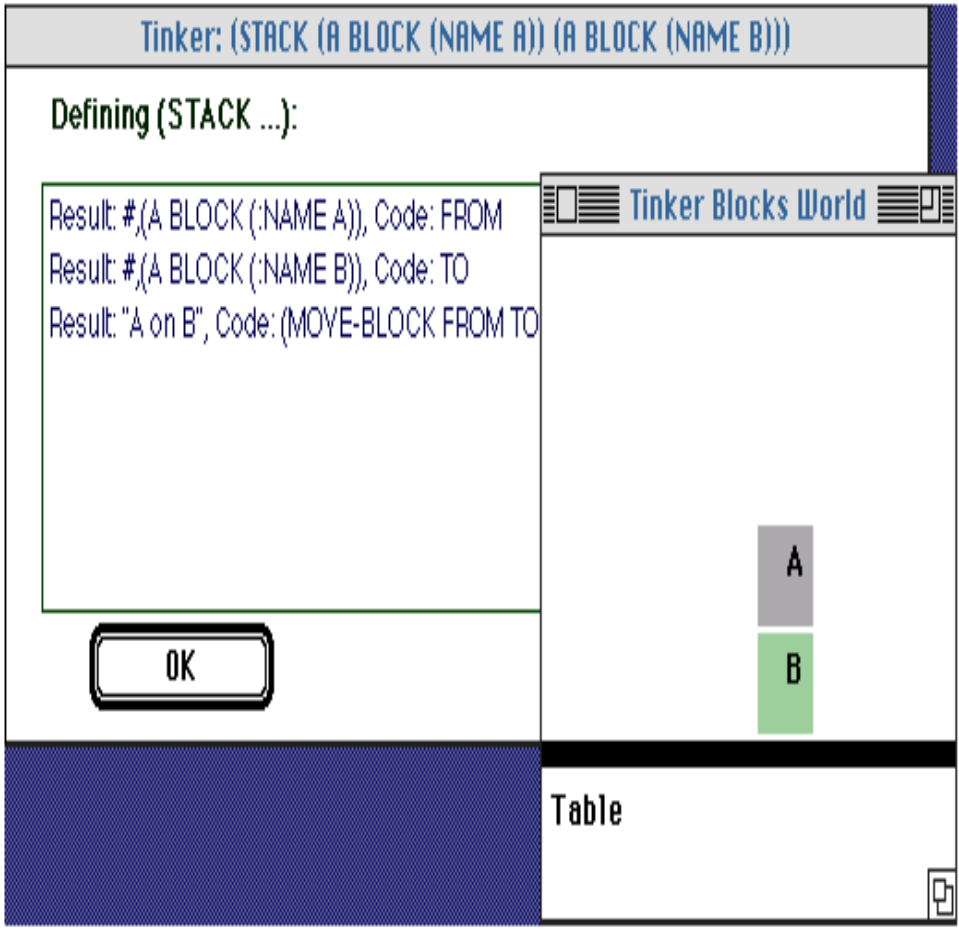


PBD. Tinker [Lieberman, 93]

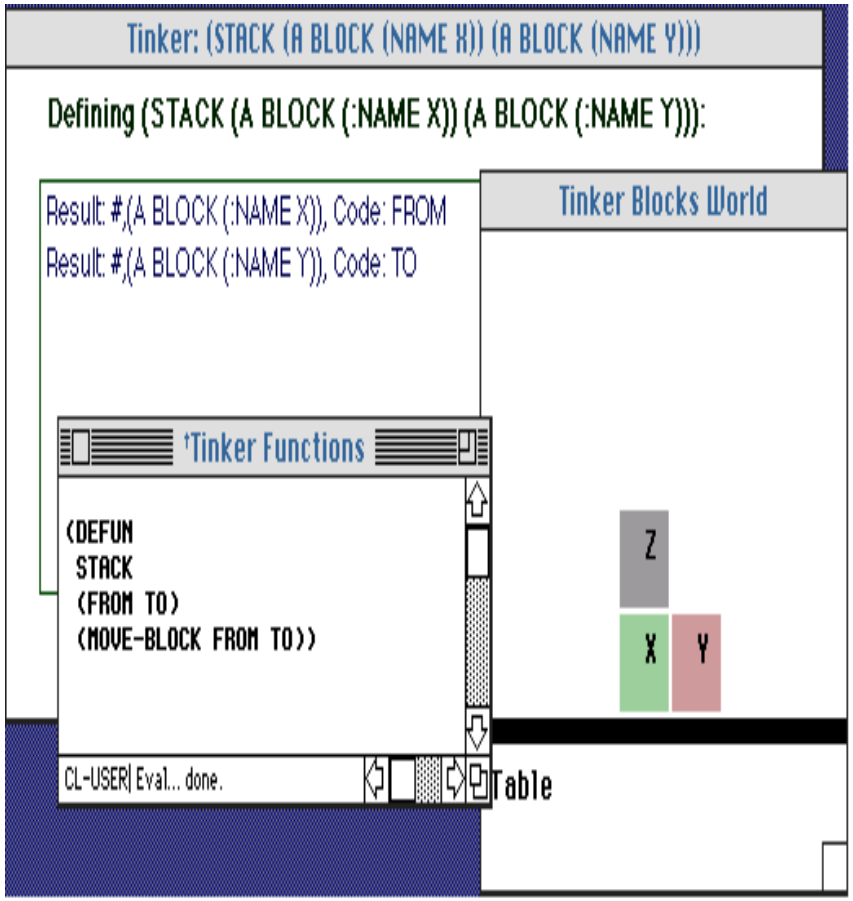
- Aprende distintas maneras de apilar un bloque sobre otro, desde la más simple a la más compleja. Interacciona con el usuario.
- Cuando detecta que para conseguir lo mismo se usan dos acciones distintas, el sistema pregunta bajo qué condiciones utilizar uno u otro (sentencia condicional)
- Muestra la importancia en la selección de casos, de lo específico a lo general y de excepciones

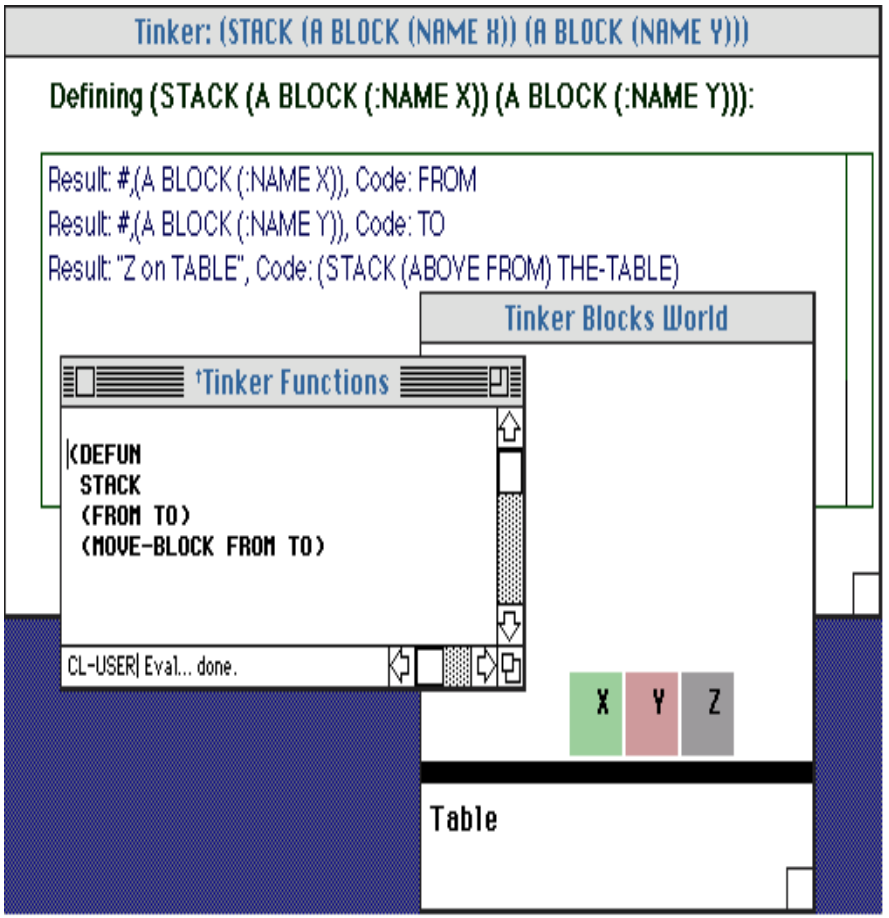
PBD. Tinker. Caso básico



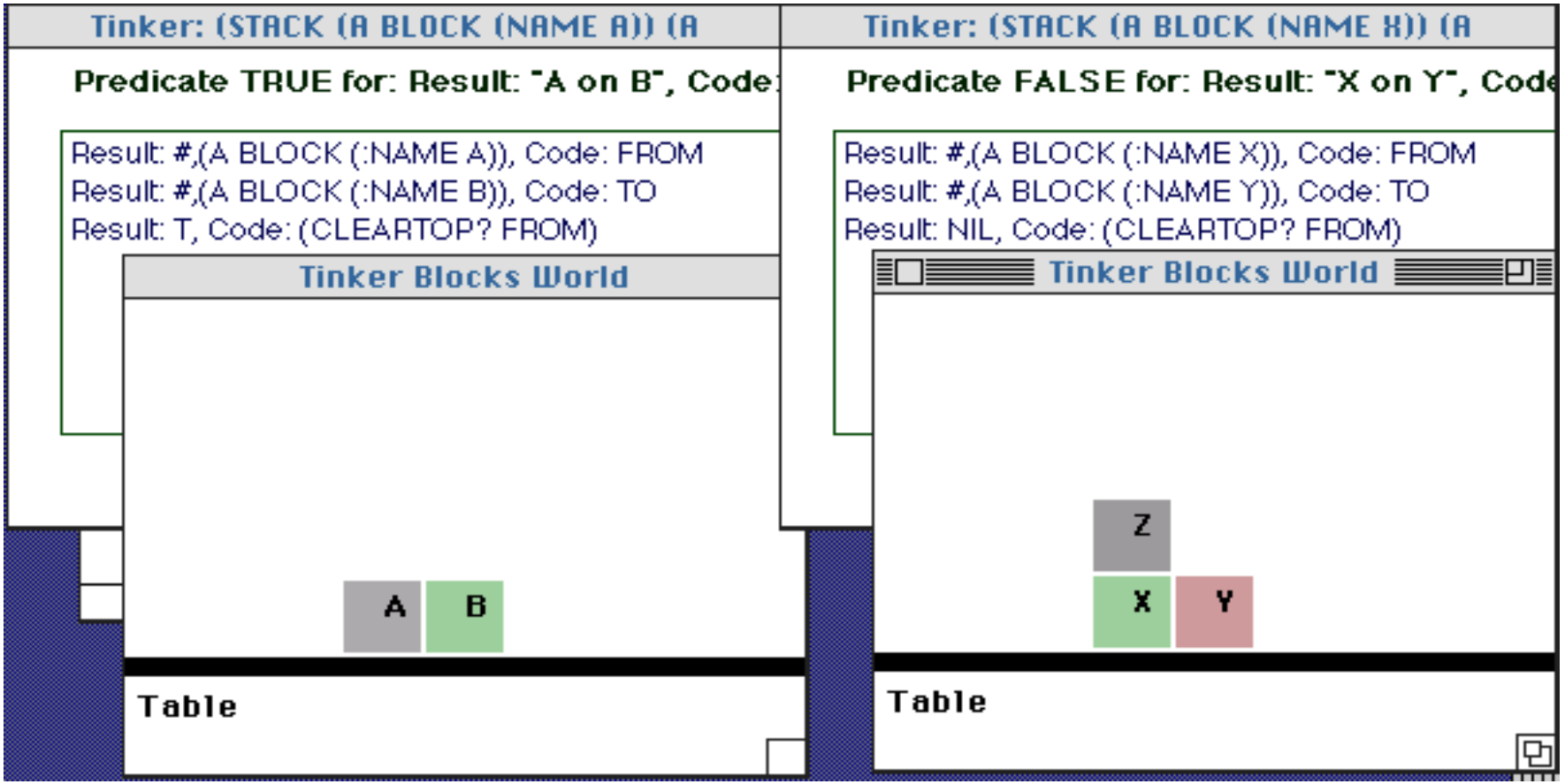


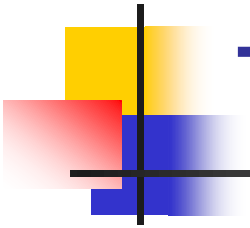
PBD. Tinker. Caso complejo





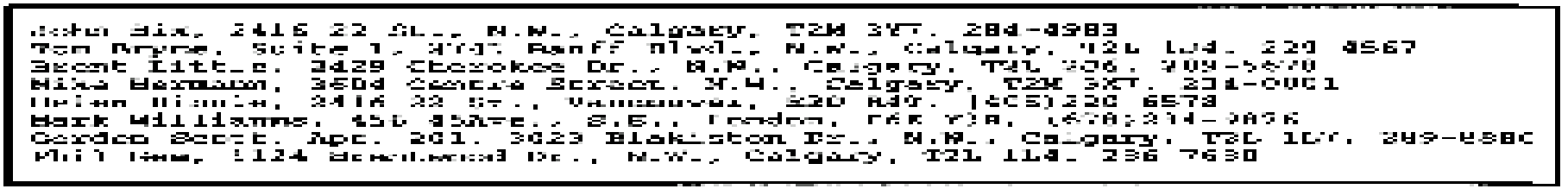
PBD. Tinker. Distinguiendo casos





TELS [Witten et al. 93]

- Cambios de formato en editores de textos
- Son tareas repetitivas
- El usuario muestra como hacer la tarea, y TELS generaliza a base de detectar repeticiones (bucles) y condicionales
- Interactivo: TELS le muestra al usuario lo que ha aprendido y este le puede corregir
- Se puede utilizar la interactividad para extender la macro a casos no vistos
- Parecido a macros, pero con generalización



Ej1: Cambio de formato en direcciones

John Rix,
2416 22 St., N.W.,
Calgary,
T2M 3Y7.

Tom Bryne,
Suite 1,
2761 Banff Blvd., N.W.,
Calgary,
T3B 1J4.

Brent Little,
2429 Cherokee Dr.,
N.W.,
Calgary,
T2L 2J5.

Mike Heimann,
3504 Centre Street,
N.W.,
Calgary,
T2M 3K7.

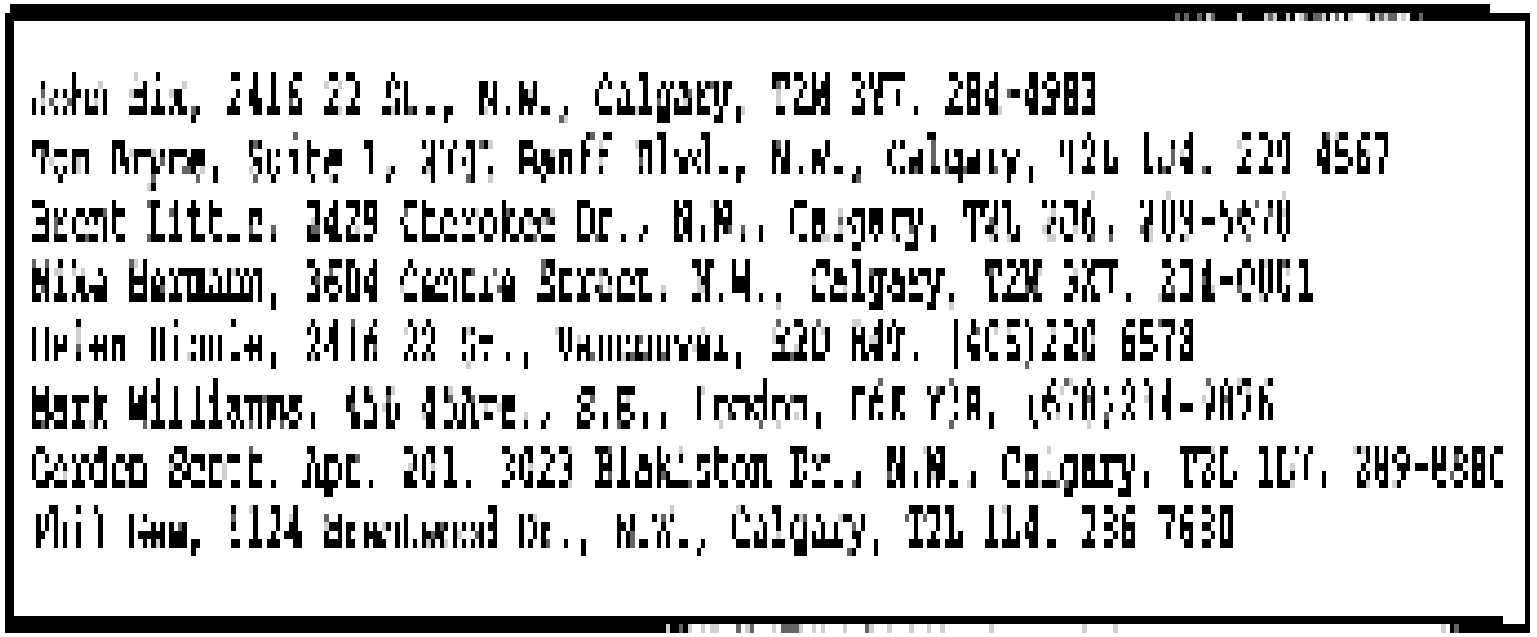
Helen Blonie,
2416 22 St.,
Vancouver,
B3D R4T.

Mark Williams,
456 45Ave., S.E.,
London,
P5E Y33.

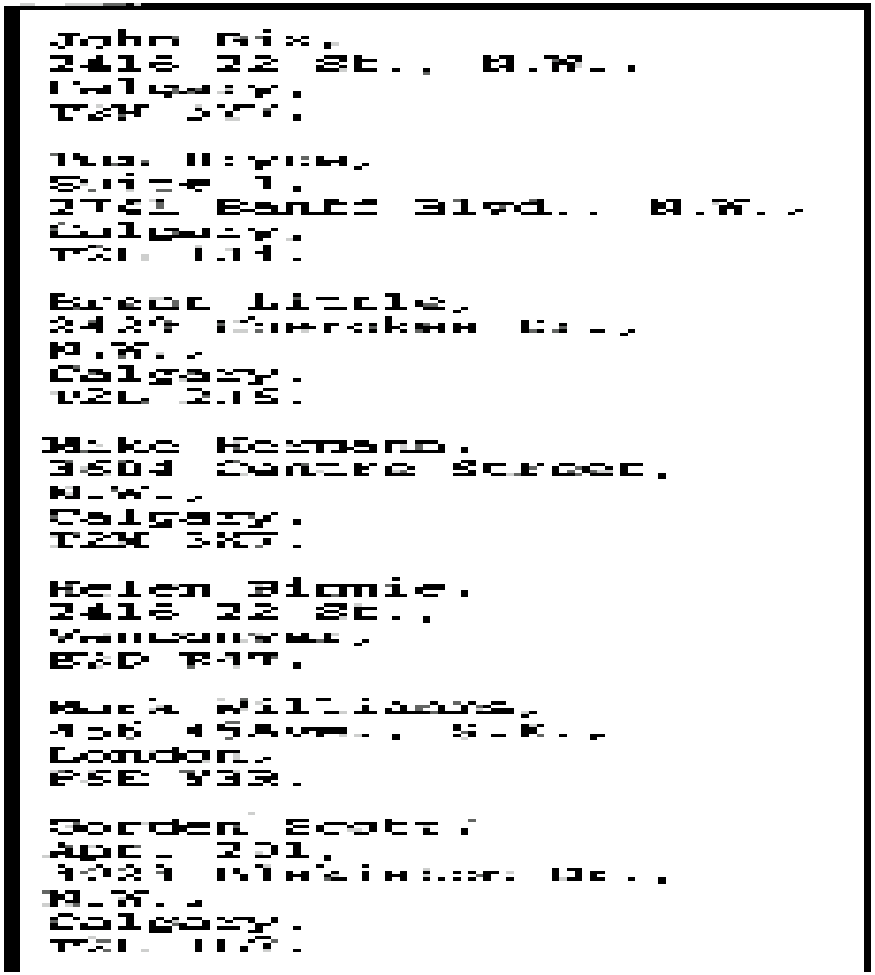
Gordon Scott,
Apt. 201,
3023 Blakiston Dr.,
N.W.,
Calgary,
T3L 1L7.

Neil Gaa,
1124 Brentwood Dr.,
N.W.,
Calgary,
T2L 1L4.

Formato de direcciones de partida

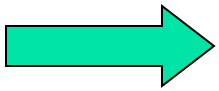


Formato direcciones de llegada



Ej2: Cambio de formato en una lista de resultados

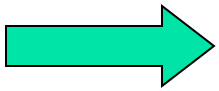
```
Cardinals 5, Pirates 2.
Tigers 3, Red Sox
1.
Red Sox 12, Orioles 4.
Yankees 7, Mets
3.
Dodgers 6, Tigers 4.
Brewers 9, Braves
3.
Phillies 2, Reds 1.
```



```
GameScore[winner 'Cardinals'; loser 'Pirates';
scores [5,2]].
GameScore[winner 'Tigers';
loser 'Red Sox'; scores
[3,1]].
GameScore[winner 'Red Sox'; loser
'Orioles'; scores [12,4]].
GameScore[winner
'Yankees'; loser 'Mets'; scores
[7,3]].
GameScore[winner 'Dodgers'; loser
'Tigers'; scores [7,3]].
GameScore[winner
'Brewers'; loser 'Braves'; scores
[9,3]].
GameScore[winner 'Phillies'; loser
'Reds'; scores [2,1]].
```

Ej3: Cambio de formato en bibliografías

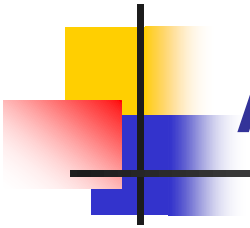
```
%A Abi-Ezzi, S.S.
%D                    *     *     * %T A
1986
%T An implementer's         framework for knowledge
view of PHIGS
%J Computer        elicita
%J Proc First
Graphics and                            European Conference on
%O
Application
%P 12-23
%O          September
%K *

%A Allen,
February
%K *

%A Ackley,         J.F.
%A Koomen, J.A.
%D
D.H.
%A Hinton, G.E.
%A          1986
%T Planning using a
Sejnowski, T.J.
%D 1985
%T       temporal world m
```



Abi-Ezzi, S.S. (1986), "An implementer's view of PHIGS", Computer Graphics and Applications, pp. 12-23, February.

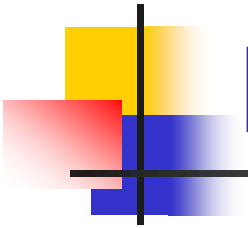
Ackley, D.H., Hinton, G.E. and Sejnowski, T.J., (1985), "A learning algorithm for Boltzmann machines", Cognitive Science, Vol. 9, pp. 147-169.

Addis, T.R. and Hinton, G.E. (1987), "A framework for knowledge elicitation", Proc First European Conference on Knowledge Acquisition, September.



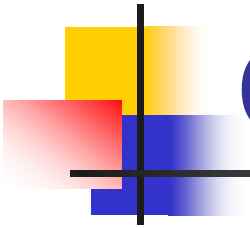
Acciones en TELS

- *Insert*(“texto”): insertar texto (*type*)
- *Locate*: colocar el ratón (*click*)
- *Select*(“texto”) seleccionar un bloque de texto
- *Delete*: borrar lo seleccionado
- Otros posibles: busca y selecciona, cut, paste, ...



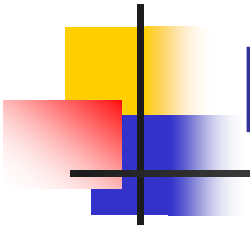
Pasos

1. Almacenar macro del usuario (secuencia de acciones del usuario)
2. Generalizarlo (variables, bucles y condicionales)
   - Encontrar pasos que se repiten: sólo se considerar pasos que tengan la misma acción
3. Ejecutarlo sobre un nuevo caso, preguntar al usuario antes de cada modificación, corregir el programa



Contexto

- Es necesario guardar el contexto para cada acción
- Por ejemplo, *locate* puede indicar distintas cosas:
  - Saltar 3 palabras desde donde estábamos
  - Colocarse al principio de palabra o de párrafo
  - Colocarse tras una coma
  - …
- Atributos para identificar el contexto:
  - Palabra anterior y posterior
  - Principio, mitad, final de línea, párrafo, fichero
  - Distancia desde la última posición: en caracteres, palabras, párrafos



Ej1: formato de trazas

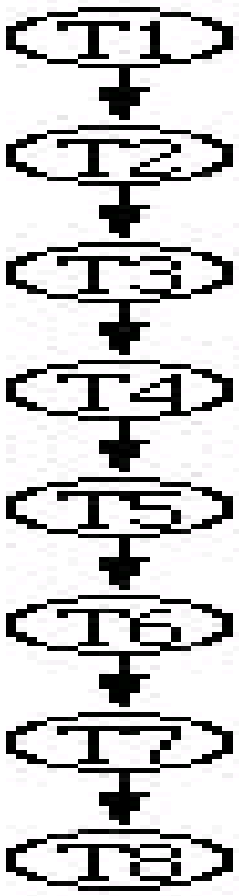
- Acción y parámetro, Contexto, Posición
- Ej: Acción y parámetro: type return
- Contexto para *locate* (antes y después):
  - N.W.,^ Calgary
- Posición:
  - num. caracteres, num. palabras, num. líneas
  - Palabra, línea, párrafo, fichero (begin, mid, end)
  - Ej: +19 +4 0 beg mid mid mid

Ej1: primer bloque

- John Blx, 24016 22 St., N.W., Calgary, T2M 3Y7. 284-4983

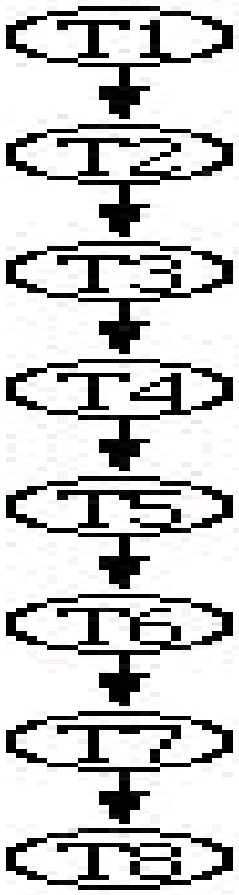
John Blx,
24016 22 St., N.W.,
Calgary,
T2M 3Y7.

Ej 1: cambio de formato en lista de direcciones



1. click Bix,^ 20416 +10 +2 0 beg mid mid mid
2. type return
3. click N.W.,^ Calgary +19 +4 0 beg mid mid mid
4. type return
5. click Calgary,^ T2M +9 +1 0 beg mid mid mid
6. type return
7. select `284-4983‘
8. type return

Ej1: secuencia aprendida



1. T1, T3, T5 se pueden considerar parte de un bucle (misma acción *click*)
2. T2, T4, T6 también (misma acción *type*)

- Se puede generalizar, siempre que se generalice el contexto de *click*
- Posible bucle:
  - Click **X**
  - Type return

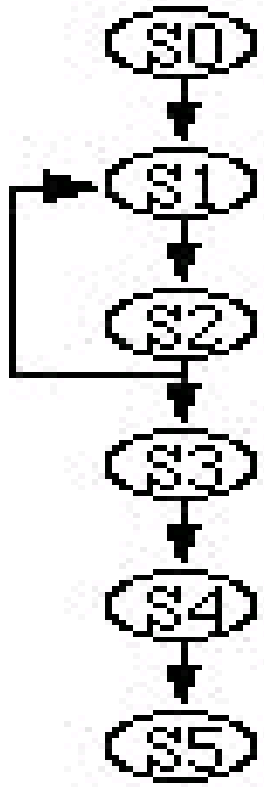
Ej1: generalización para unir T1, T3, T5

- T1: *click* **Bix,^ 2416** +10 +2 0 beg mid mid mid
- T3: *click* **N.W.,^ Calgary** +19 +4 0 beg mid mid mid
- T5: *click* **Calgary,^ T2M** +9 +1 0 beg mid mid mid
- S1: *click* [c]*,^ [a]* ? ? 0 beg mid mid mid

Es decir, hay que hacer un click entre “[c]*” y “[a]*”, en la misma línea y al comienzo de una palabra

Nota: [c] significa carácter, [a] significa alfanumérico

Ej1: detección de bucle



Start

click [c]*,^ [a]* 0 beg mid mid mid

type return

select ‘284-4983’

type return

Stop

Ej1: segundo bloque

- Tom Bryce, Suite 1, 2741 Baniff Blvd., N.W., Calgary, T2L L24. 229-4567

El usuario acepta (de manera interactiva) los dos primeros cambios, pero rechaza el tercero (el sistema ha sobregeneralizado)

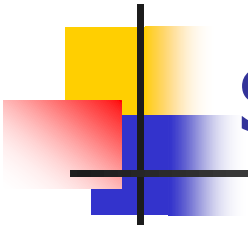
Resultado obtenido

Tom Bryce,
Suite 1,
2741 Baniff Blvd.,
N.W.

Resultado deseado

Tom Bryce,
Suite 1,
2741 Baniff Blvd., N.W.,
Calgary,
T2L L24.

Ej1: reparación de la sobregeneralización



- Se guarda un ejemplo negativo, para que no parta la línea en ese sitio:
  - S1: Blvd.,^ N.W.

  (volverá a fallar en el siguiente bloque de texto)
- Se especializa el contexto para que no devuelva éxito en esa posición (en este caso, [c]*,^ [a]* no se puede especializar más)

Ej1: segundo bloque

- Tom Bryce, Suite 1, 2741 Baniff Blvd., N.W., Calgary, T2L L24. 229-4567

Resultado obtenido

Tom Bryce
Suite 1
2741 Baniff Blvd.,N.W.
Calgary
T2L L24.

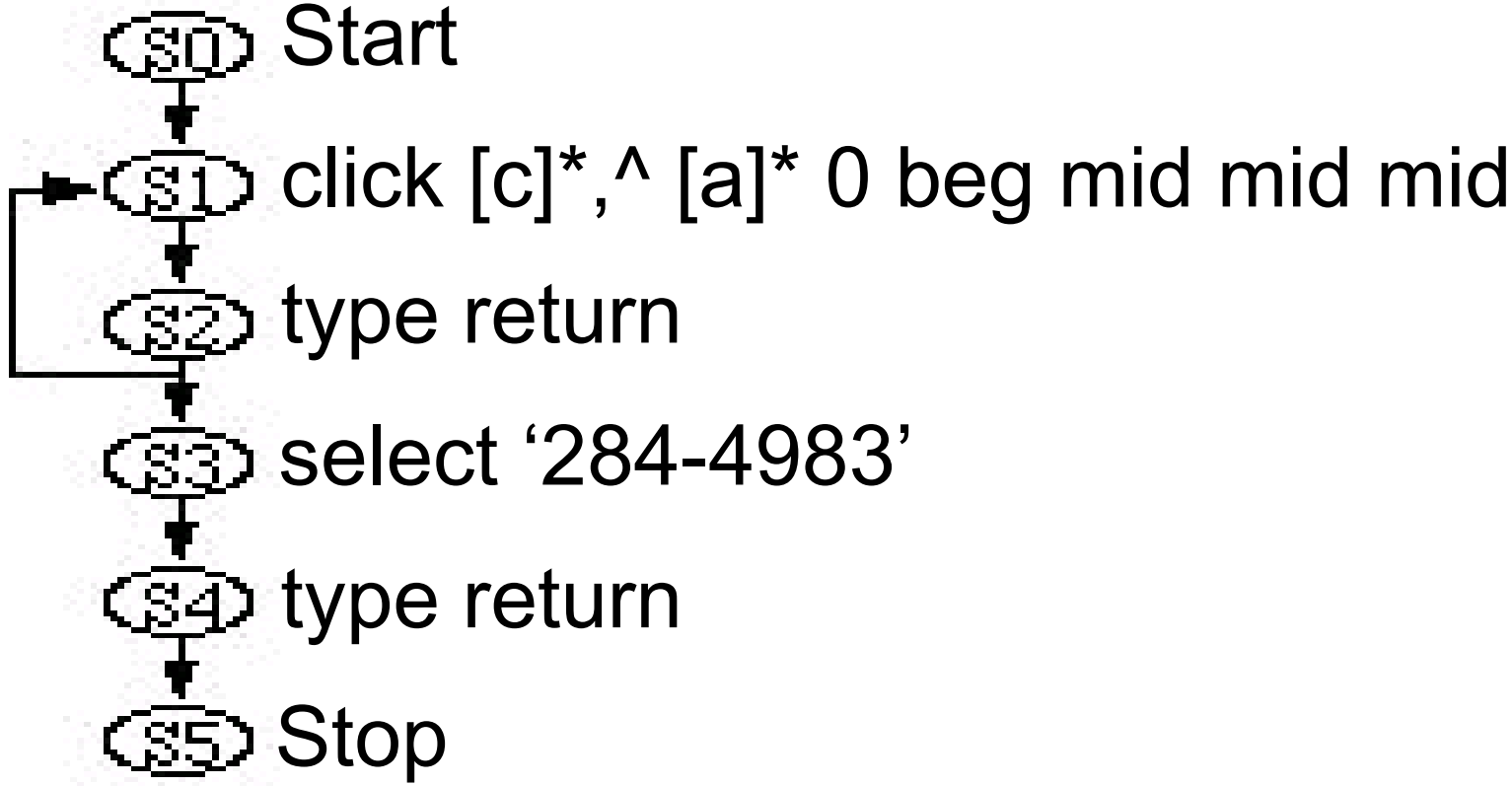
Resultado deseado

Tom Bryce
Suite 1
2741 Baniff Blvd., N.W.
Calgary
T2L L24.

Ej 1: fallo en S1 y S3

Ni las condiciones de S1 o S3 funcionan con 229-4567.

Hay que generalizar el argumento de select

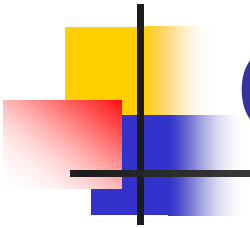


Ej 1: generalización de *select*

- Select ‘284-4983’
- Select ‘229-4567’

Se generaliza a:

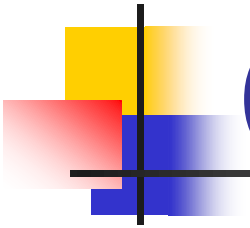
- Select "2[digit]* -4[digit]*"
- Este funciona y el usuario acepta la predicción
- Caso de no funcionar, otras generalizaciones:
  - “[digit]*[op][digit]*"
  - "[alfanum]*[no-alfanum][alfanum]*"



Generalización

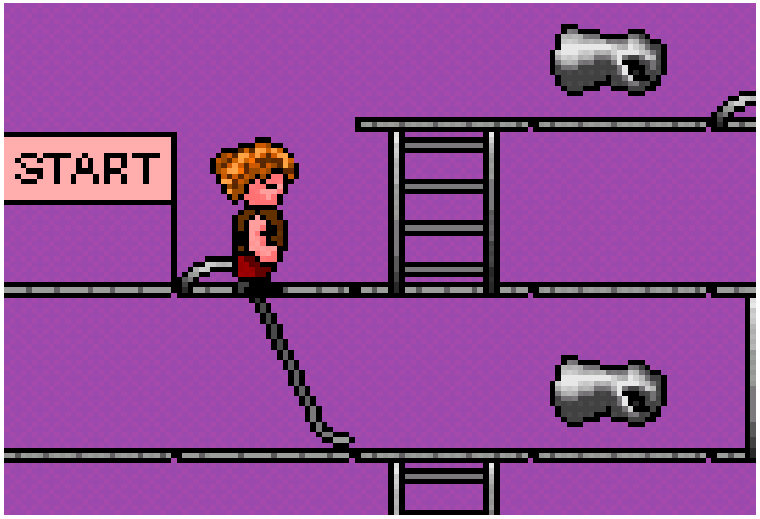
- De localizadores:
  - Begin, end -> extreme
  - Begin, middle -> ?
  - 3, 4 -> ?
  - 3, 3 -> 3
- De cadenas de caracteres: encontrar subsecuencias constantes y generalizar el resto:
  - ‘284-4983’, ‘229-4567’ -> "2[digit]*-4[digit]*"

PBD. Otras aplicaciones (macros generalizan)



- **Predictive calculator** (observa al usuario teclear y predice operaciones)
- **Smallstar:** aprende a hacer tareas compuestas en el desktop. Repeticiones y condicionales se añaden a mano
- **Eager:** Aprende a detectar acciones repetitivas. Predice la siguiente y el usuario confirma o rechaza
- Predicción de URLs, generación de Web wrappers, ...

PBD. Stagecast (creación de juegos y simulaciones con agentes y reglas)

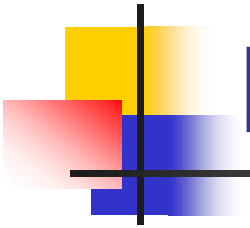


http://www.stagecast.com/creator.html

0:36 Simulaciones life-cycle, difusion

2: Creación de caracteres y reglas

7: Juego de coches



PBD. Conclusiones

- En la práctica, PBD se centra en la obtención de macros generalizadas
- Dominios específicos, relativamente sencillos (no es programación general)
- Dificultad: **es difícil capturar las intenciones del usuario** por mera observación (ej: en "uno, dos, **tres**" ¿el usuario quiere marcar la tercera palabra, o una palabra que empieza por t?
- Buenas ideas:
  - Utilización de trazas proporcionadas por el usuario
  - Posibilidad de preguntar e interaccionar con el usuario