

4

Planificació i anàlisi econòmic del projecte

En aquest capítol es fa un anàlisi de la planificació temporal del projecte. S'analitzen les diferents etapes principals en el desenvolupament del projecte, fent una comparativa entre el temps que s'havia previst emprar en la consecució d'aquesta etapa i el temps que realment s'ha necessitat finalment.

A continuació es fa també un anàlisi de despeses econòmics del projecte, tant pel que fa a les hores de treball personal com per les eines utilitzades, tant de programari com de maquinari.

4.1 Planificació temporal

El desenvolupament del projecte s'ha dut a terme en diverses etapes. Inicialment es va fer una previsió temporal de les hores necessidades per a cada una d'aquestes etapes, previsió que no s'ha acomplert per la dificultat que representa fer aquesta mena de planificació la primera vegada que es du a terme un projecte d'aquesta magnitud. Les etapes en les que s'ha dividit el projecte són les següents:

- Etapa 1. Estudi teòric breu sobre les cascades de molècules.
- Etapa 2. Anàlisi de requeriments.
- Etapa 3. Especificació.
- Etapa 4. Disseny.
- Etapa 5. Estudi de les possibles tecnologies a utilitzar per a la implementació.
- Etapa 6. Estudi i proves de la biblioteca wxWidgets.
- Etapa 7. Creació de l'entorn de desenvolupament a Sourceforge.
- Etapa 8. Implementació.
- Etapa 9. Proves de correctesa del programa.
- Etapa 10. Redacció de la memòria.
- Etapa 11. Redacció del manual d'usuari.

Etapa	Hores planificades	Hores reals
1	5	5
2	5	10
3	80	70
4	80	60
5	5	5
6	20	10
7	5	10
8	350	450
9	50	40
10	150	180
11	2	5
Total	752	845

Taula 4.1: Planificació temporal.

A la taula 4.1 es pot observar la planificació de cada una de les etapes. S'hi pot veure les hores que es van planificar inicialment i, en segon lloc, les hores que realment han calgut per completar-les.

Com podem observar hi ha una desviació total de 93 hores de més. Aquesta desviació ve donada principalment en l'etapa d'implementació, en la que s'han dedicat 100 hores més de les inicialment planificades. La causa d'aquest desviament és la clara subestimació que es va fer inicialment de, primerament, la quantitat de coses a fer, i en segon lloc la sèrie de dificultats que s'han trobat per la no excessiva experiència en la programació de projectes d'aquesta magnitud.

La memòria també ha costat unes quantes hores més d'escriure, especialment per les contínues revisions i modificacions que se n'han fet.

També és curiós veure com, en les etapes referents a l'Enginyeria del Software, l'especificació i el disseny, s'han dedicat menys hores de les inicialment planificades.

4.2 Anàlisi econòmic

Per fer un anàlisi econòmic tindrem en compte dos factors:

- Les hores de treball.
- Eines utilitzades.

Pel que fa a les hores de treball, s'han agrupat diverses etapes en funció del nivell de valoració laboral que aquestes tenen. Així doncs, separarem les hores d'analista del que serien les hores de programador. A la categoria d'anàlisi hi posarem les etapes 1, 2, 3, 4 i 10. Les etapes 5, 6, 7, 8, 9 i 11 seran considerades hores de programador.

La taula 4.1 mostra la planificació temporal.

Pel que fa a les eines, la taula 4.3 mostra les despeses necessàries del projecte.

El cost total del projecte és de 25206€.

Etapa	Hores	Preu per hora	Preu total
1	5	36€	180€
2	10	36€	360€
3	70	36€	2520€
4	60	36€	2160€
5	5	24€	120€
6	10	24€	240€
7	10	24€	240€
8	450	24€	10800€
9	40	24€	960€
10	180	36€	6480€
11	5	24€	120€
Total		24180€	

Taula 4.2: Cost de les hores de treball.

Eina	Preu
Ordinador domèstic	1000€
Llibre sobre wxWidgets	26€
Sistema operatiu Linux	gratuït
Compilador gcc	gratuït
Software d'enginyeria del software Umbrello	gratuït
Infraestructura Sourceforge	gratuït
Generador de documentació Doxygen	gratuït
IDE Eclipse	gratuït
Total	1026€

Taula 4.3: Cost de les eines utilitzades.

5

Conclusions

5.1 Objectius assolits

Un cop finalitzat el projecte estem en condicions d'assegurar que tots els objectius del projecte han estat assolits.

El programa ha estat desenvolupat en la seva totalitat i implementa totes les funcionalitats que s'havia acordat en l'anàlisi de requisits.

A més del propi desenvolupament del programa, s'ha dut a terme un profund procés de documentació de tot el codi font. Aquesta documentació permetrà amb molta més facilitat futures modificacions o ampliacions del programa per afegir-hi noves funcionalitats, ja que és una eina bàsica per entendre el funcionament del software.

També s'ha redactat un petit manual d'usuari que permetrà començar a usar el programa a aquells que desconequin el seu funcionament, tot i que el programa és força intuitiu.

5.2 Treball futur

Hi ha certs detalls que poden treballar-se en futures revisions del programa i que, tot i que no estaven incloses en les especificacions inicials, seria interessant estudiar-les.

5.2.1 Estudi de la portabilitat

Tot i que tant C++ com wxWidgets són un llenguatge de programació i una biblioteca portables i multiplataforma, molt sovint aquestes característiques no són tan certes com voldriem.

El programa ha estat desenvolupat i provat extensivament en un sistema operatiu Linux, el compilador GCC i les biblioteques GTK+.

Seria interessant fer un estudi de la portabilitat del programa a altres platformes, concretament a Mac OS i a Windows, per veure si realment el programa més portable al 100% o bé cal fer lleugeres modificacions per assegurar el seu funcionament en aquests sistemes.

5.2.2 Estudi de l'eficiència

Tot i que les graelles amb les que treballa el programa difícilment superaran un tamany de 20x20, com s'ha comentat en el disseny del sistema software, s'ha utilitzat un algorisme *naïve* per al *pattern matching* en dues dimensions. Seria interessant fer un petit anàlisi de l'eficiència d'aquest algorisme i la programació d'algorismes alternatius per comparar-ne resultats.

De manera similar, el widget que dibuixa les taules de veritat funciona de manera extraordinàriament lenta quan ha de pintar taules de deu o més entrades. S'ha analitzat aquest problema i s'ha vist que rau en la classe `wxGrid` que s'utilitza com a base. Seria interessant estudiar on és el problema, i si calgués fer un disseny i implementació més eficients d'una classe totalment personalitzada.

5.3 Conclusions

El balanç del projecte en la seva totalitat ha estat molt positiu.

El desenvolupament en la seva totalitat d'un sistema software no trivial m'ha donat l'oportunitat d'aprendre moltes coses, i sobretot d'adonar-me de la utilitat de moltes de les assignatures que he anat cursant durant la carrera, que és un dels objectius principals del desenvolupament del Projecte de Final de Carrera.

En primer lloc he apès la vital importància de fer un bon anàlisi de requisits. Tot i que el projecte ja es va plantejar inicialment com un sistema que s'hauria d'anar modificant i adaptant a les circumstàncies, és absolutament vital deixar clar, des d'un principi, què és allò que el client vol del sistema a desenvolupar. Deixar aquest anàlisi de requisits per escrit per tenir, en un futur, una eina alhora de justificar que realment s'ha fet el treball que es va demanar i res més.

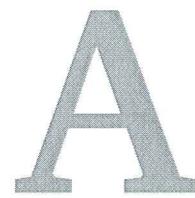
Per altra banda, i seguint amb la branca de l'Enginyeria del Software, la importància fonamental del procés d'especificació i disseny dut a terme abans de començar amb la implementació. Evidentment, cap disseny és perfecte, i hi ha certes parts que s'han hagut de reespecificar i redissenyar a posteriori, però això ja entra dins el concepte de cicle de vida en espiral, on noves situacions o fets no tinguts en compte en el passat provoquen haver de tornar enrera i fer més feina de la que inicialment s'havia planificat.

Pel que fa a la implementació, el desenvolupament d'aquest projecte m'ha proporcionat experiència tant en C++ com en wxWidgets. Tot i que ja havia fet algun altre programa amb C++ —sense tenir en compte algunes de les pràctiques de la carrera, d'un abast clarament inferior al d'aquest projecte—, el desenvolupament de *nanocomp* m'ha permès aprendre coses del llenguatge que no sabia i, sobretot, a no cometre errors típics de programació amb els que m'he anat trobant eventualment.

Haver après wxWidgets és també quelcom que trobo molt positiu. No només perquè realment no s'ensenya pràcticament res sobre GUIs a la carrera sinó pel fet que, essent multiplataforma, és un recurs al que sempre podré recórrer en cas que desitgi desenvolupar algun altre programa gràfic tant en Windows, Linux, com a Mac OS.

He après també a utilitzar amb profunditat el debugger `gdb`, ja que he hagut de resoldre una quantitat bastant elevada de bugs que han anat apareixent amb relativa constància durant tot el programa, i a causa també de l'ús de diferents versions del compilador `gcc`, que donava resultats diferents en algunes crides i que m'han dut força mals de cap.

Voldria, per últim, remarcar l'esforç i temps dedicat a una documentació exhaustiva i completa de tot el codi font del programa. He hagut de treballar amb anterioritat amb programes ja fets per altres persones i malauradament m'he trobat, més sovint del que voldria, amb codi font sense documentar. Conscient del que m'ha costat entendre els programes que he hagut de modificar en aquestes condicions, he intentat documentar el programa pensant precisament en aquesta circumstància.



Documentació

La documentació online del programa està disponible en format HTML online a la següent URL: <http://nanocomp.sourceforge.net/documentation/>. Hi trobareu documentació relativa a les classes, funcions i crides del programa. És una documentació extraordinàriament útil per entendre com està programat el software, i per dur a terme posteriors modificacions. La documentació ha estat creada utilitzant el sistema Doxygen i no s'ha inclòs en aquesta memòria per qüestions d'espai, ocupant aquesta més d'un centenar de pàgines, i també perquè, per la seva naturalesa, el format HTML és el més apropiat per visualitzarla.

La documentació està creada en anglès per motius pràctics, per si alguna persona estrangera desitgés modificar el programa i ajustar-lo a les seves necessitats.

B

User manual

We'll see a full example of a molecule cascade simulation. The logic gate we'll implement is a simple fork. A fork is a duplication of an input into more than one outputs.

The first thing we should do is start the program. You should see a window like the one in figure B.1.

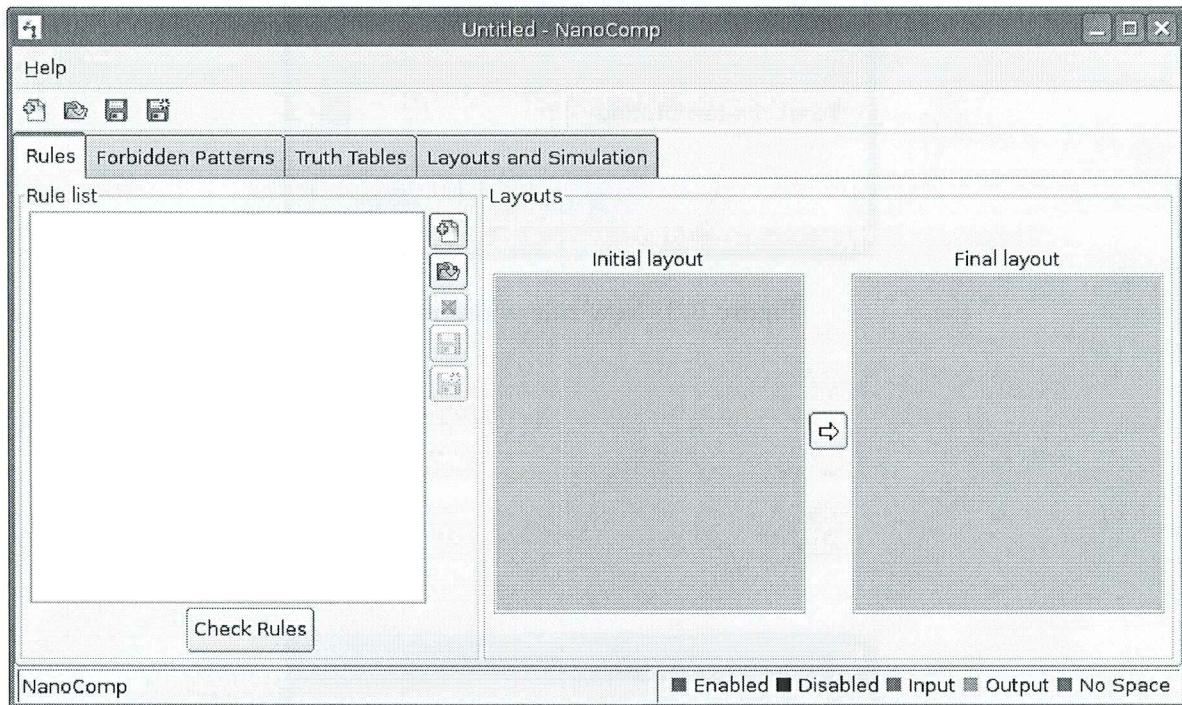


Figure B.1: NanoComp main window

First of all, we will create the main rule that the simulation will use. Rules are pairs of configurations. When simulating, if the initial configuration is found on the space, the rule is applied, and the configuration becomes the final one.

To create a rule, you must click on the "Add a new rule button" (figure B.2). After that, you must enter the width and the height of the rule. We will create a typical rule in molecule cascades, a chevron. The chevron is a 3x3 rule, so we will input a width and height values of 3, as shown in figures B.3 and B.4.

After that, an empty rule will be created. Our job is to modify it to meet our requirements.



Figure B.2: “New” button.

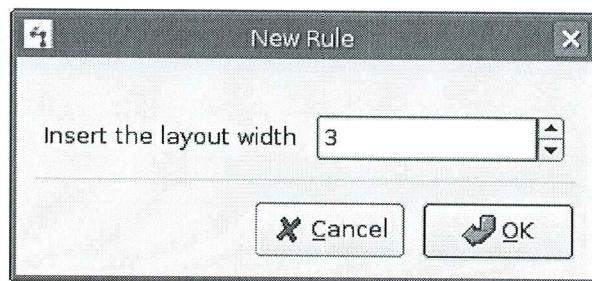


Figure B.3: New rule width input.

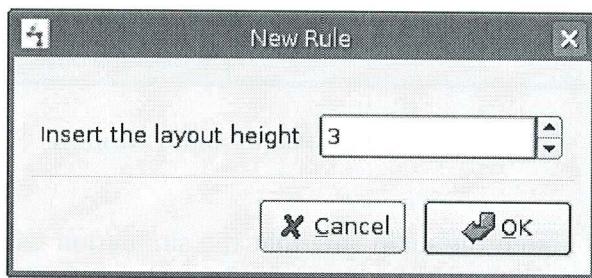


Figure B.4: New rule height input.

Modifying the rule is as simple as clicking on the cell point we want to change. The initial rule will be shown as in figure B.5.

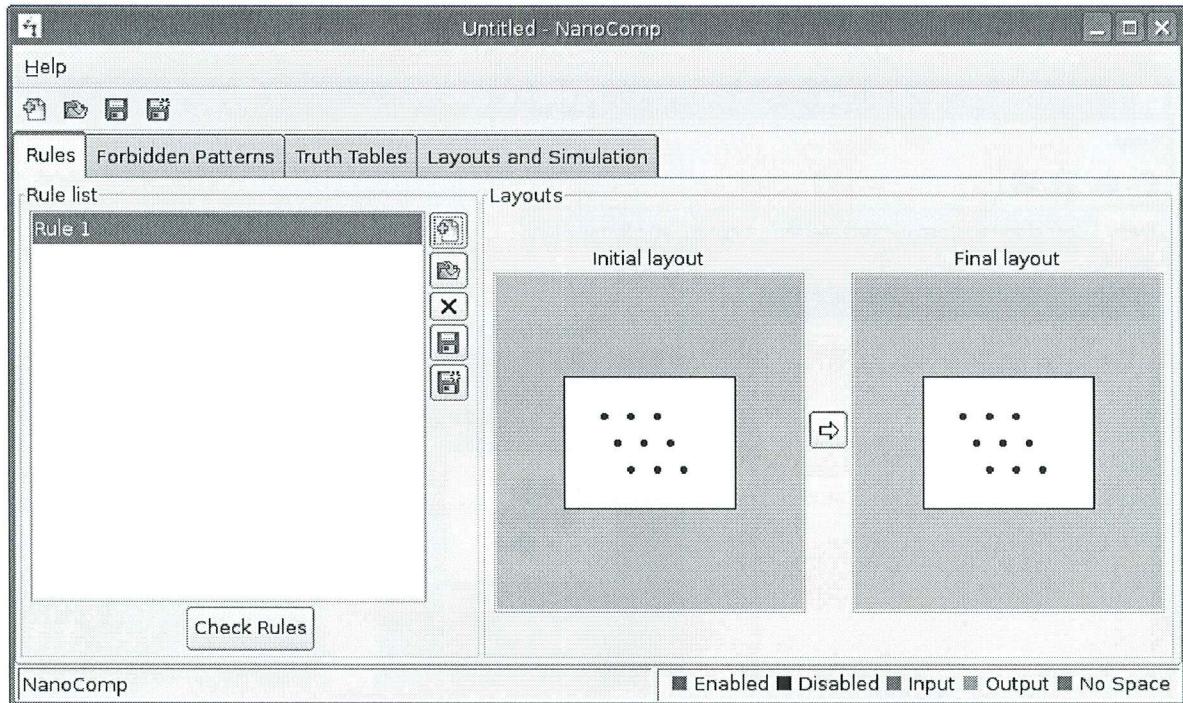


Figure B.5: New rule created.

As you can see, we have two 3x3 grids. Those little points are empty cells. To modify them, we must click on them. Its status will change with every click, so we must click the cell as many times as necessary until it has the status we want.

We will start by modifying the left grid. This is the initial configuration of the rule, and the pattern that will be searched in the simulation. Click on each of the cells until you have the image shown in figure B.6.

Now we have two ways to configure the right grid. We could just do the same we've already done, but there is a faster way: clicking on the arrow button (figure B.7) all the cells on the left configuration will be copied to the right one, so we will only have to modify the cells that actually differ from configurations. Using the method you like, modify the final configuration until you have the result shown in figure B.8:

Now we should create the forbidden pattern. To do so, we must click on the "Forbidden Patterns" tab. We will see a screen like the one in figure B.9.

Just like we did for the rules, create a new forbidden pattern of size 3x3 and change its cells until you get what is shown in figure B.10.

Time to configure the truth table. Click on the "Truth Tables" tab and then on the new truth table button. This time you'll be asked for three parameters rather than two: the table name, the table number of inputs and the table number of outputs. As we are building a fork table, you can name the table "Fork". The table will have one input and two outputs, which will have to be the same value as the input, as shown in figures B.11, B.12 and ??.

The table will be created and you'll see a screen like the one in figure B.14.

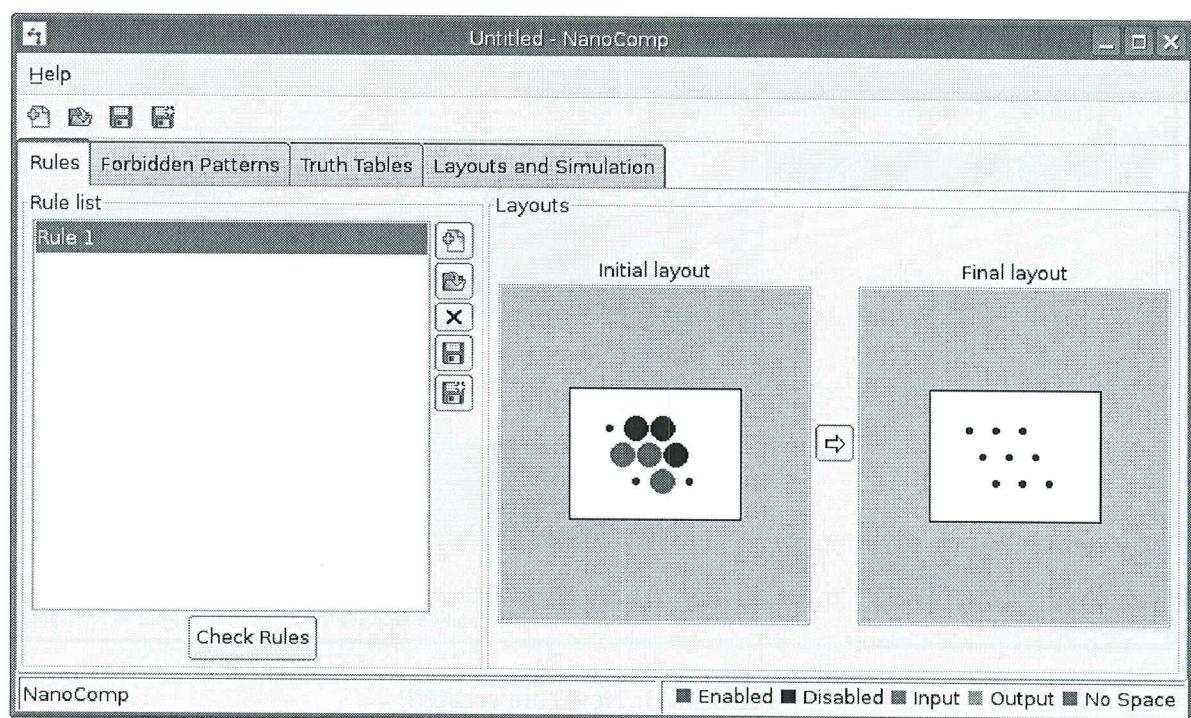


Figure B.6: Initial configuration of the rule modified.



Figure B.7: “Copy rule” button.

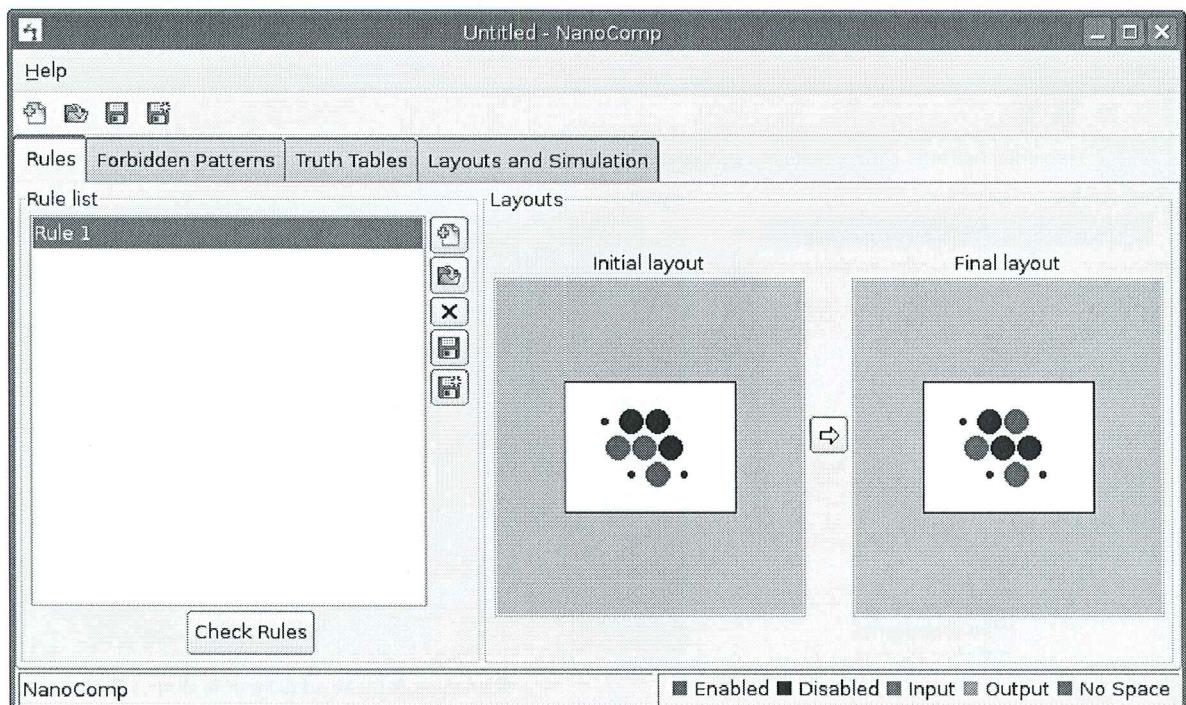


Figure B.8: Rule fully modified.

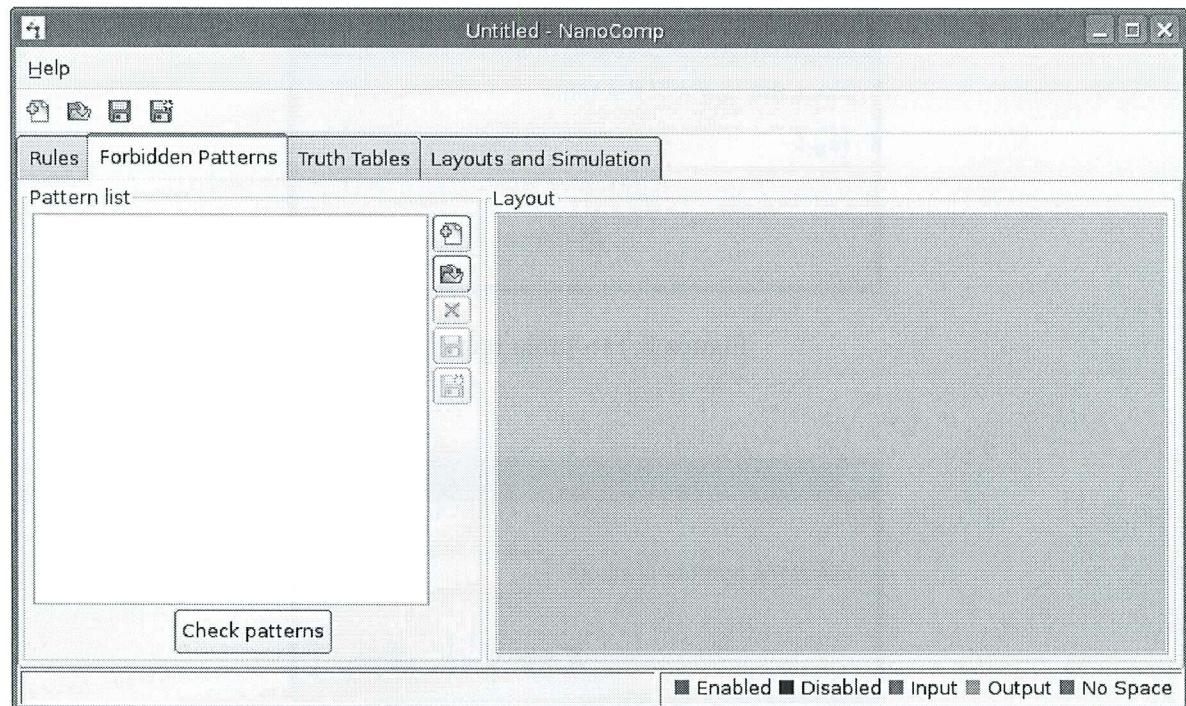


Figure B.9: Forbidden pattern view.

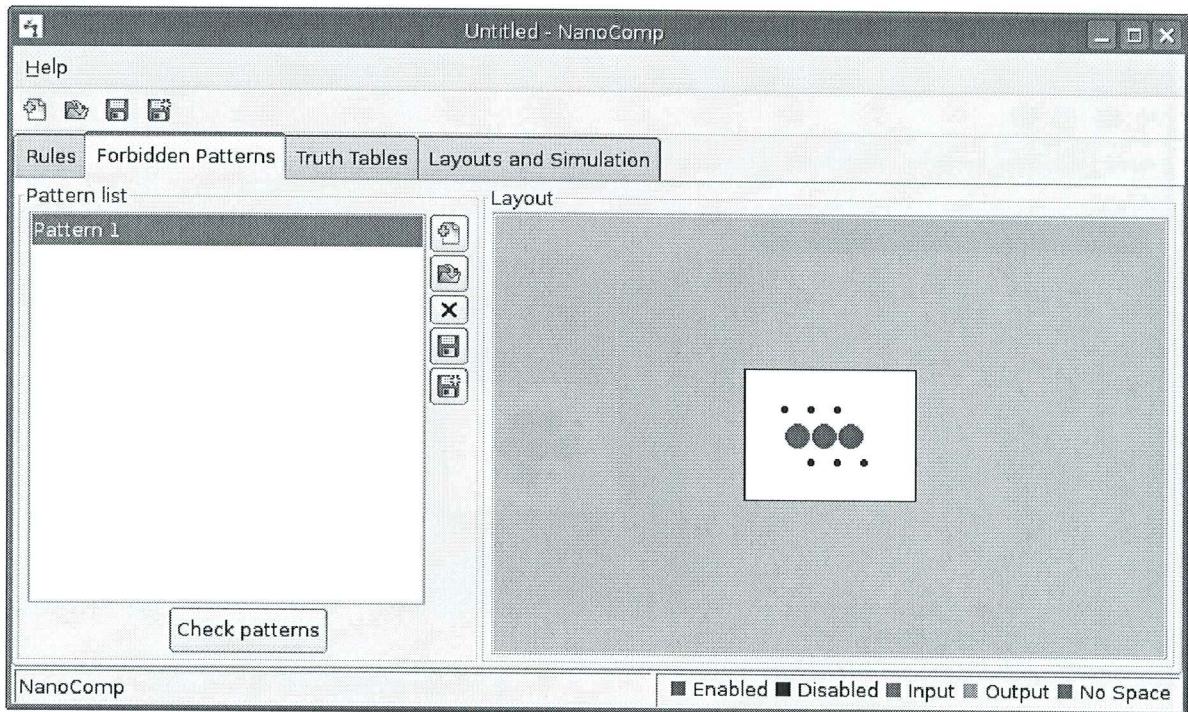


Figure B.10: Forbidden pattern modified.

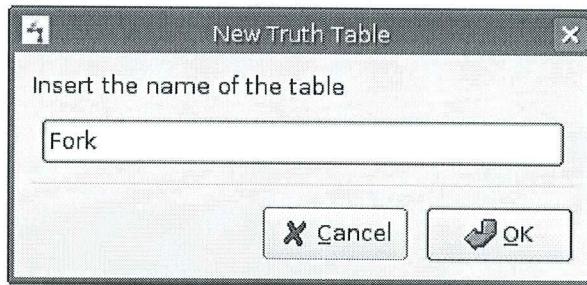


Figure B.11: Table name.

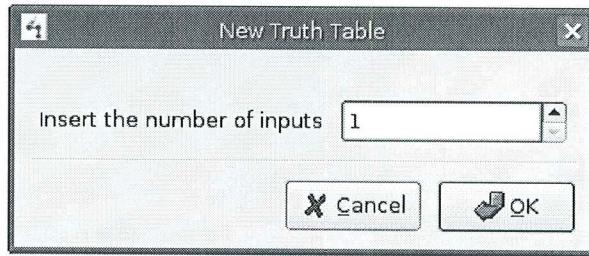


Figure B.12: Table inputs.

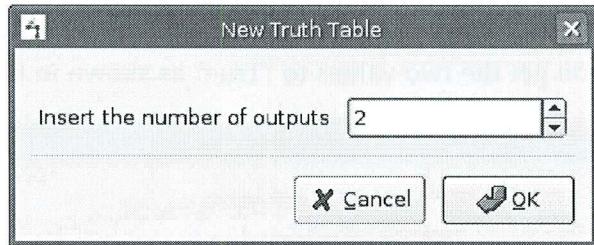


Figure B.13: Table outputs.

A screenshot of the NanoComp software interface, titled "Untitled - NanoComp". The window has a menu bar with "Help" and several icons. Below the menu is a toolbar with icons for file operations. The main area has tabs: "Rules", "Forbidden Patterns", "Truth Tables" (which is selected), and "Layouts and Simulation". On the left, there's a "Truth Table list" containing a single item named "Fork". The central panel is titled "Truth Table" and displays a 2x4 grid. The columns are labeled "Input 1", "Output 1", and "Output 2". The rows are labeled "1" and "2". The values in the grid are: Row 1, Input 1: 1, Output 1: False, Output 2: False; Row 2, Input 1: 2, Output 1: True, Output 2: False. A tooltip at the bottom of the grid says "Click on an output value to change it". At the bottom of the window is a legend: "Enabled" (green square), "Disabled" (red square), "Input" (blue square), "Output" (orange square), and "No Space" (grey square).

Figure B.14: Table created.

As you can see, the table input combinations are already calculated by the program. You only have to worry to configure the correct outputs for each of the possible input combination. As we've said, this is a fork, so the output values should match the input values. That means that the two "False" values from the 2nd row must be changed to "True". The way to do this is the same as we did for the cells: just click on the cell you want to change and voilà... it'll change. Do this until you get the two values to "True" as shown in figure B.15.

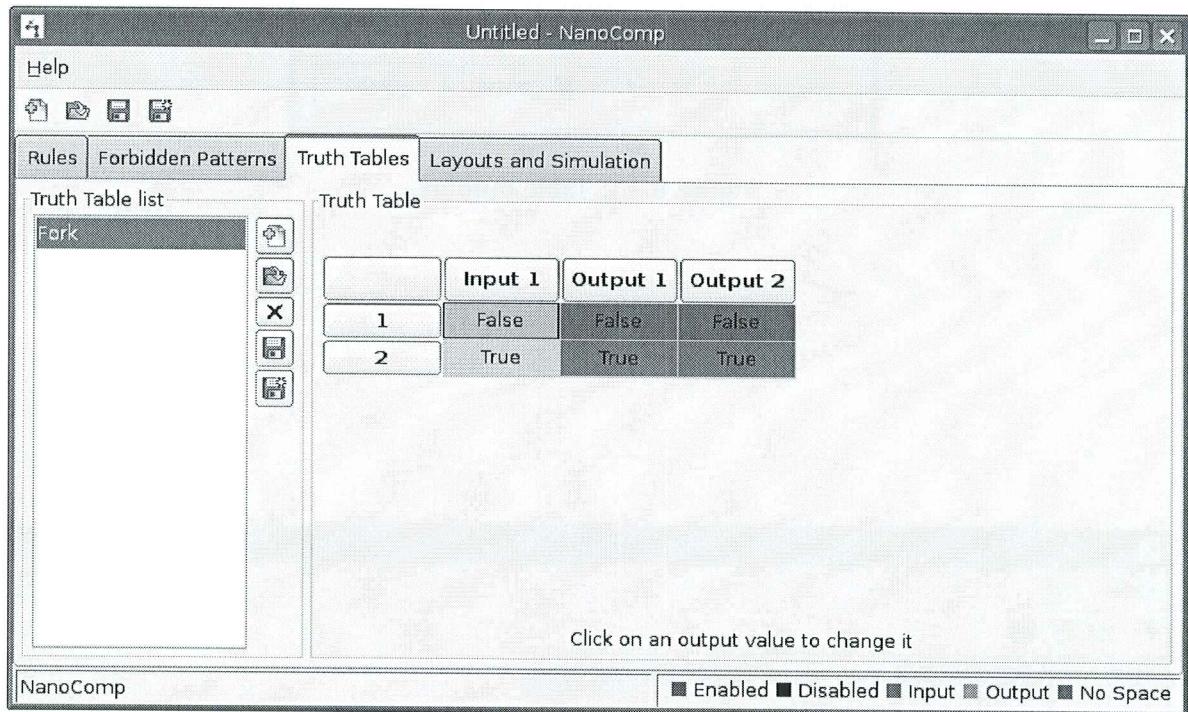


Figure B.15: Table modified.

And now for the final configuration tab, the "Layouts and Simulation" tab. Click on the new Space button and select a 5x5 space. Select 1 input and 2 outputs, the same we did when creating the table. Now change the space cells to fit those shown in figure B.16.

As you can see in the figure, the Fork table checkbox is checked. Check it in your window too. This check will ensure that the fork table is the one we are going to simulate. In the same way, the rule and pattern we created will be automatically checked. Uncheck them if you don't want them to be applied at the simulation.

Now that we have the space created and the table checked, there is only one thing left to do: assign the inputs and outputs to the space. To do this, you have to select the input/output you want to assign and then click on the corresponding button, in figure B.17. When you do this, the window will change as shown in figure B.18 (for an input).

At this moment, only the input cells will be available to click. When you click on the cell, the input selected will be assigned there. If you want to go back and not assign the input, just press the escape key.

Now, assign the input and the outputs. As we only have an input and both outputs share the values, its not important where you assign in. The final window should look similar to figure B.19.

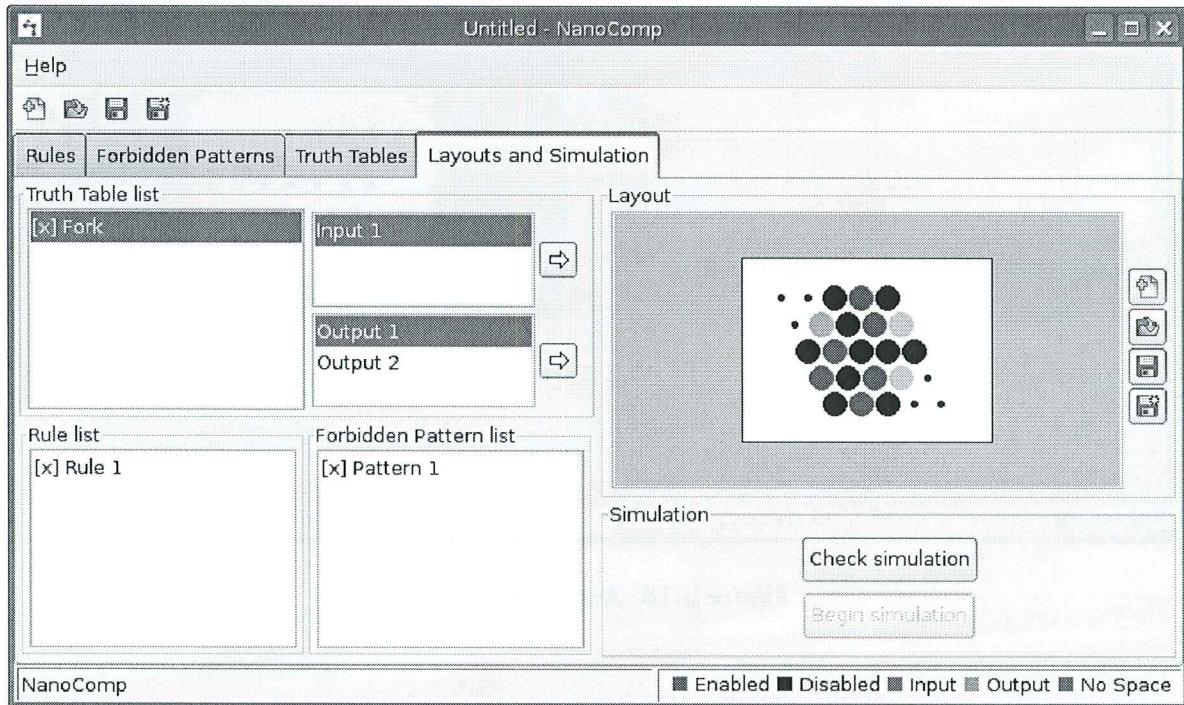


Figure B.16: Space tab.



Figure B.17: “Assign” button.

Apèndix B. User manual

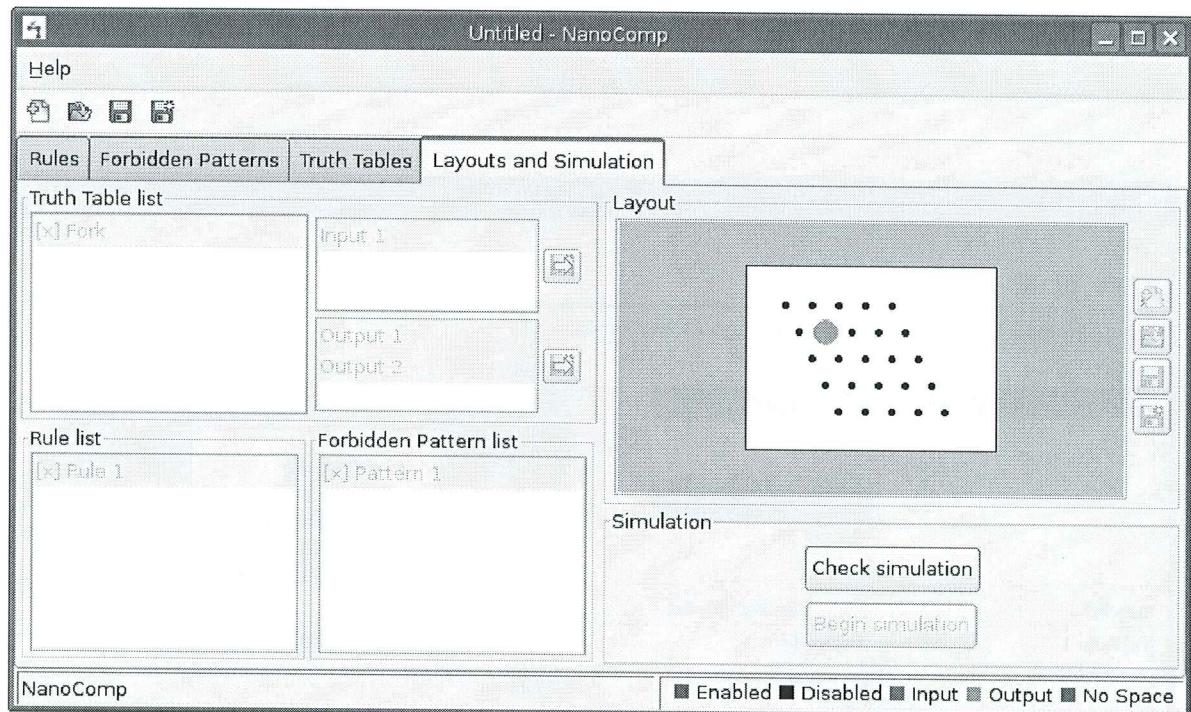


Figure B.18: Assigning an input.

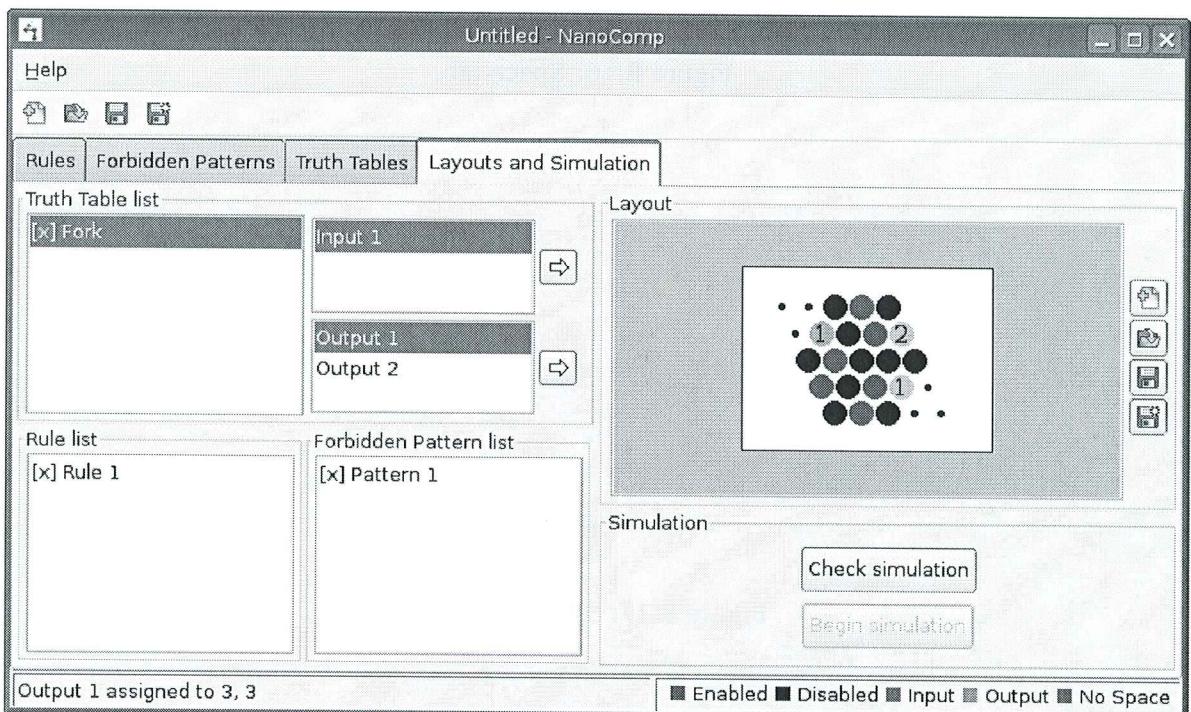


Figure B.19: Space configured.

Now we are ready to simulate. Clicking on the “Check simulation” button will check everything to ensure the simulation can start. Click it, and then click on the “Begin simulation” button to actually start the simulation. You’ll see the image on figure B.20.

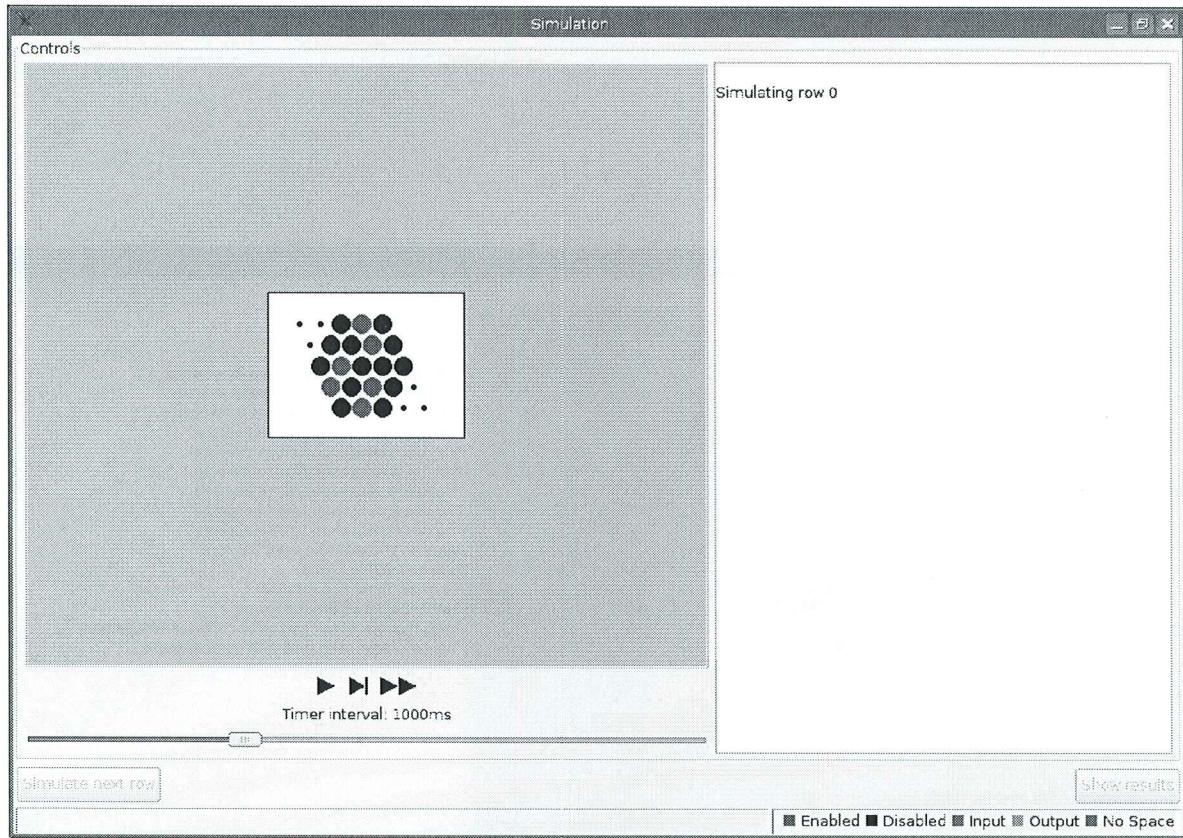


Figure B.20: Simulation window.

You can simulate in three different ways:

- Automated simulation
- Step by step simulation
- All in one step simulation

The automated simulation will execute simulation steps automatically every second. You’ll have to manually change rows, though. It’s started and paused clicking on the left button.

The step by step simulation is fully manual. You have to execute the steps manually by clicking the middle button.

The all in one step simulation will execute all the steps of every table row without any king of user interaction. It’s started with the right button.

To see the simulator in action, press the automated simulation button and watch. When a row finishes its execution, the simulation will stop and ask you to simulate the next row; press the “Simulate next row” to do so (figure B.21).

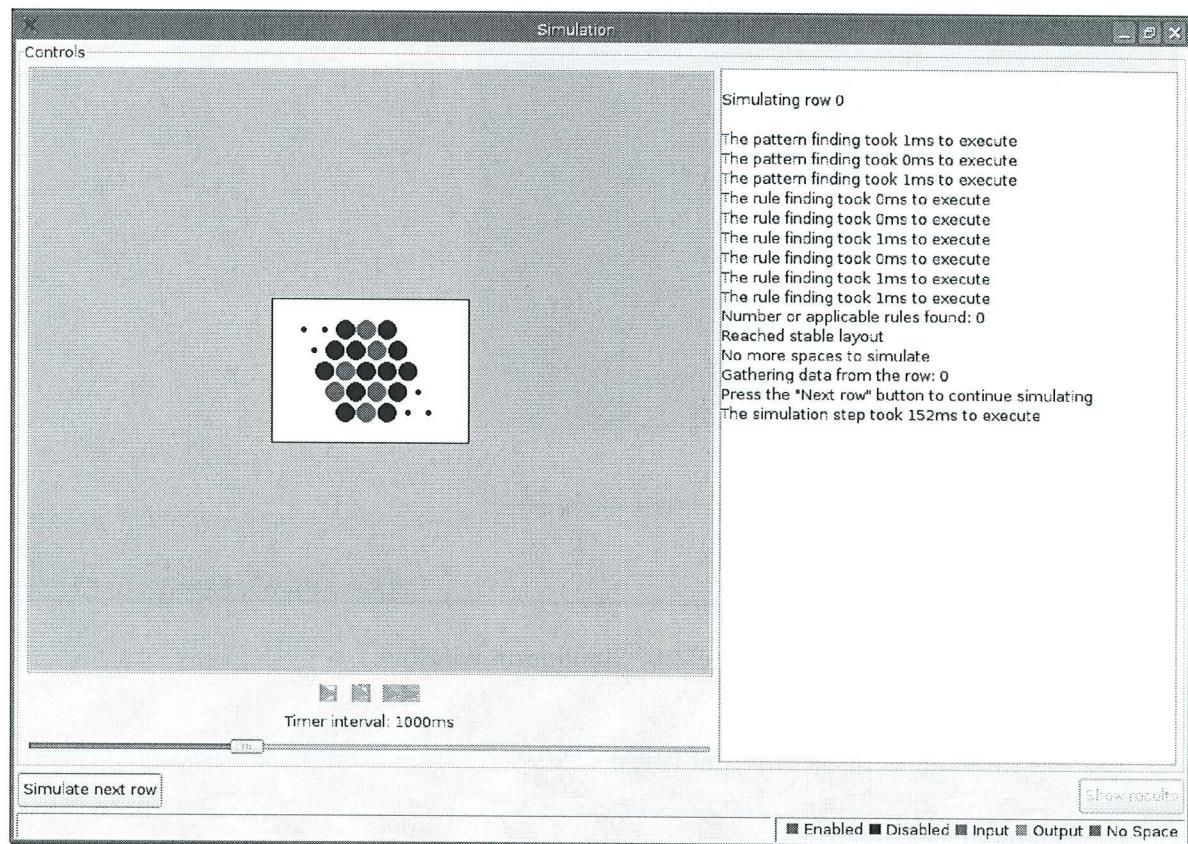


Figure B.21: Row simulation finished.

After that, press the automated simulation button again to begin the simulation of the new row.

After all rows have been simulated, the “Show results” button will be enabled; click on it to see the results of the simulation (figure B.22).

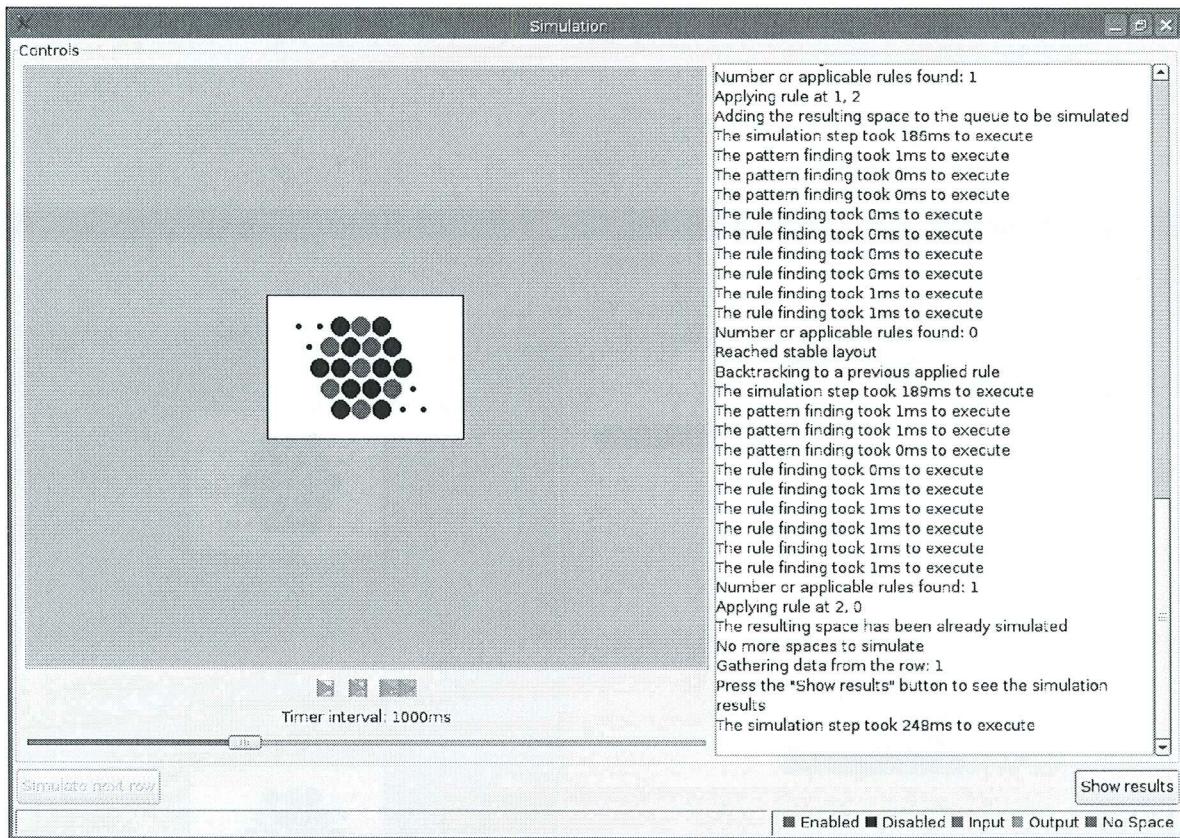


Figure B.22: Simulation finished.

After pressing the button, the results view will be shown on the screen (figure B.23). The results view shows four quarters:

- The simulation information provides information about the simulation steps of every row of the table.
- The truth table provides the truth table that has been simulated.
- The simulation step shows the simulation step selected in the simulation information.
- The initial configuration shows the initial space of the simulation with its inputs and outputs.
- Only the simulation information quarter is interactive. The results of what you choose there will be shown in the simulation step.

The simulation information provides three columns:

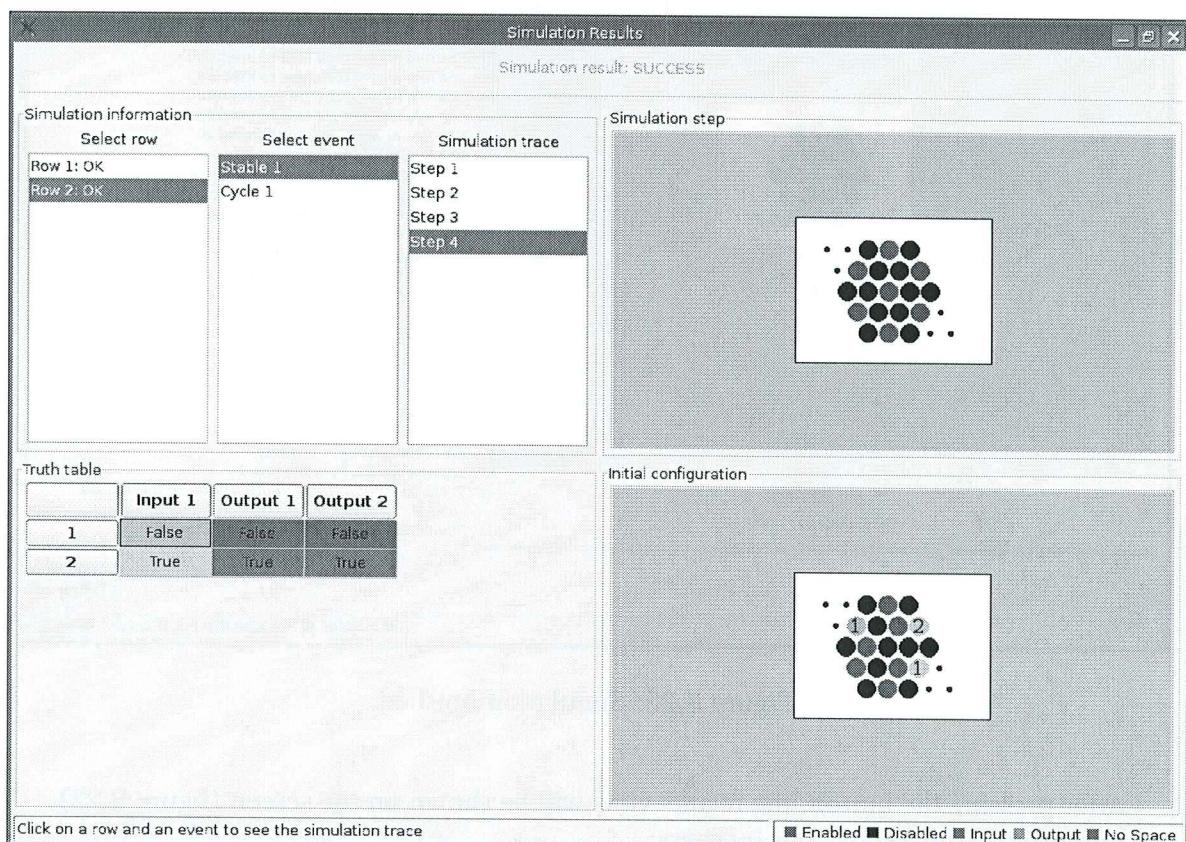


Figure B.23: Simulation results.

-
- The first column contains all the rows of the table. Showing if they were verified or not. Choosing one of the rows, the information on the middle column will be updated.
 - The middle column contains all the final status the simulation has reached for the row. If it found a forbidden pattern or an out of bounds rule, only that information will be shown. If not, it will show all the stable spaces and cycles reached by the simulation. Selecting one of the rows of that list, the right column will be updated.
 - The last column contains all the simulation trace that lead to the information you selected on the middle column. By selecting any of the steps, the space of the step will be shown in the simulation step quarter.

This ends the example tutorial. For information about all the remaining features visit the online documentation at <http://nanocomp.sourceforge.net/manual>.

C

Eines utilitzades

nanocomp és un programa desenvolupat sota la llicència GNU GPL¹. A més, totes les eines utilitzades en el procés de desenvolupament d'aquest han estat també eines lliures.

El compilador utilitzat ha estat *gcc* (<http://gcc.gnu.org/>), tant la nova versió 4.0 com la versió més antiga 3.3. No s'ha provat el programa amb altres compiladors.

Com a IDE² s'ha decidit utilitzar *Eclipse* (<http://www.eclipse.org/>), per la seva facilitat d'ús, característiques i gran integració amb l'eina CVS.

Tota la part referent a l'enginyeria del software, creació de diagrames d'estats, de classe i de seqüència, s'ha fet amb l'eina *Umbrello* (<http://uml.sourceforge.net/>), integrada dins la suite de desenvolupament de l'escriptori *KDE*³ de Linux.

La memòria que estàs llegint en aquests moments ha estat creada utilitzant la coneguda eina de creació de textos científics *LATEX* (<http://www.latex-project.org/>).

La documentació del codi font s'ha fet amb *Doxxygen* (<http://www.stack.nl/~dimitri/doxygen/>), eina que genera automàticament documentació en diversos formats, posant comentaris al codi font seguint un determinat format.

Tota la part referent a la creació gràfica s'ha desenvolupat amb l'eina *GIMP* (<http://www.gimp.org/>).

Finalment, el projecte s'ha dut a terme amb el recolzament de la comunitat Sourceforge⁴. Sourceforge és una comunitat online de recolzament a projectes GNU. Entre les diferents facilitats que proporciona als desenvolupadors hi ha un servei de CVS i un espai web. L'ús del CVS ha estat de gran utilitat perquè s'ha treballat en aquest projectes des d'ordinadors diferents, i el CVS ha permès una sincronització sense problemes entre ordinadors. A més, en tot moment el director del projecte ha pogut baixar del CVS l'última versió desenvolupada per poder-la provar.

¹General Public License

²Integrated Development Environment

³K Desktop Environment

⁴<http://sourceforge.net>

D

Obtenció del codi i instal·lació

El codi del programa es pot obtenir de dues maneres diferents:

- Descarregant la distribució de la web.
- Descarregant la versió de desenvolupament del CVS.

En funció de quina versió es descarregui el procés d'instal·lació varia lleugerament.

D.1 Obtenció del fitxer de distribució

Per descarregar el codi de la pàgina web, s'ha d'accedir a l'URL <http://nanocomp.sourceforge.net>, on es troba un enllaç al codi font comprimit en format tar.gz.

El fitxer s'ha de descomprimir utilitzant la comanda tar xvzf <nom del fitxer>. Aquesta comanda crearà un directori anomenat nanocomp-<versió>.

Un cop descomprimit el codi font, hem d'entrar en aquest directori i executar les comandes següents:

1. ./configure

2. make

3. make install

L'última comanda s'ha d'executar amb permisos de root.

Un cop fet això tindrem el programa ja instal·lat. Executant la comanda nanocomp arrenarem el programa.

D.2 Obtenció de la versió de desenvolupament del CVS

Per descarregar la versió de desenvolupament del CVS s'haurà d'executar la comanda següent:
cvs -z3 -d:pserver:anonymous@nanocomp.cvs.sourceforge.net:/cvsroot/nanocomp co -P nanocomp.

Un cop executada aquesta comanda tindrem un directori anomenat nanocomp. Hem d'entrar en aquest directori i executar les comandes següents:

1. ./bootstrap.sh

2. ./configure

3. `make`

4. `make install`

Un cop fet això tindrem el programa ja instal·lat. Executant la comanda `nanocomp` arrenarem el programa.

Bibliografia

- [AJHE02] J. A. Gupta A. J. Heinrich, C. P. Lutz and D. M. Eigler. Molecule cascades. *Science*, 298:1381–1387, 2002.
- [BYR93] Ricardo A. Baeza-Yates and Mireille Regnier. Fast two-dimensional pattern matching. *Information Processing Letters*, 45(1):51–57, 1993.
- [CGT03] Antoni Olivé Cristina Gómez, Enric Mayol and Ernest Teniente. *Enginyeria del software Disseny I*. Edicions UPC, 2003.
- [DCT00] M. Ribera Sancho Dolors Costal and Ernest Teniente. *Enginyeria del software Especificació*. Edicions UPC, 2000.
- [DMEG04] M. F. Crommie H. C. Manoharan A. J. Heinrich D. M. Eigler, C. P. Lutz and J. A. Gupta. Information transport and computation in nanometre-scale structures. *Phil. Trans. R. Soc. Lond. A*, 362(1819):1135 – 1147, 2004.
- [FF04] Eric Freeman and Elisabeth Freeman. *Head First Design Patterns*. O'Reilly Media, 2004.
- [JSC05] Kevin Hock Julian Smart and Stefan Csomor. *Cross-Platform GUI Programming with wxWidgets*. Prentice Hall Professional Technical Reference, 2005.
- [Str98] Bjarne Stroustrup. *El lenguaje de programación C++*. Turpial, 1998.

