Version

1

BCN Protoype User's Manual

not!

AN IMPLEMENTATION OF CONSTRUCTIVE NEGATION
FOR NORMAL LOGIC PROGRAMS

BCN Prototype Version 1.0.1

J. Álvez
P. Lucio

Universidad del País Vasco / Euskal Herriko Unibertsitatea

F. Orejas
E. Pasarella
E. Pino

Universidad Politécnica de Catalunya

This work has been partially supported by the Spanish Project TIC 2001-2476-C03

Chapter

1

INTRODUCTION

- The BCN Prototype is a sound and complete implementation of constructive negation for the whole class of normal logic programs.
- It has been implemented in SICStus Prolog 3.8.5.
- The theoretical foundations of this implementation come from Shepherdson's operators.
- These operators provide a bottom-up scheme for computing literal answers, that is the basis of our implementation.
- The constraint solver that we have implemented obtains the literal answers in an efficient, incremental and lazy way.
- The procedural mechanism for goal computation, not only obtains all the correct answers for a given goal, but also detects failure. In spite of the bottom-up nature of the answers calculation, the procedural mechanism is in charge of detecting when a goal should fail.

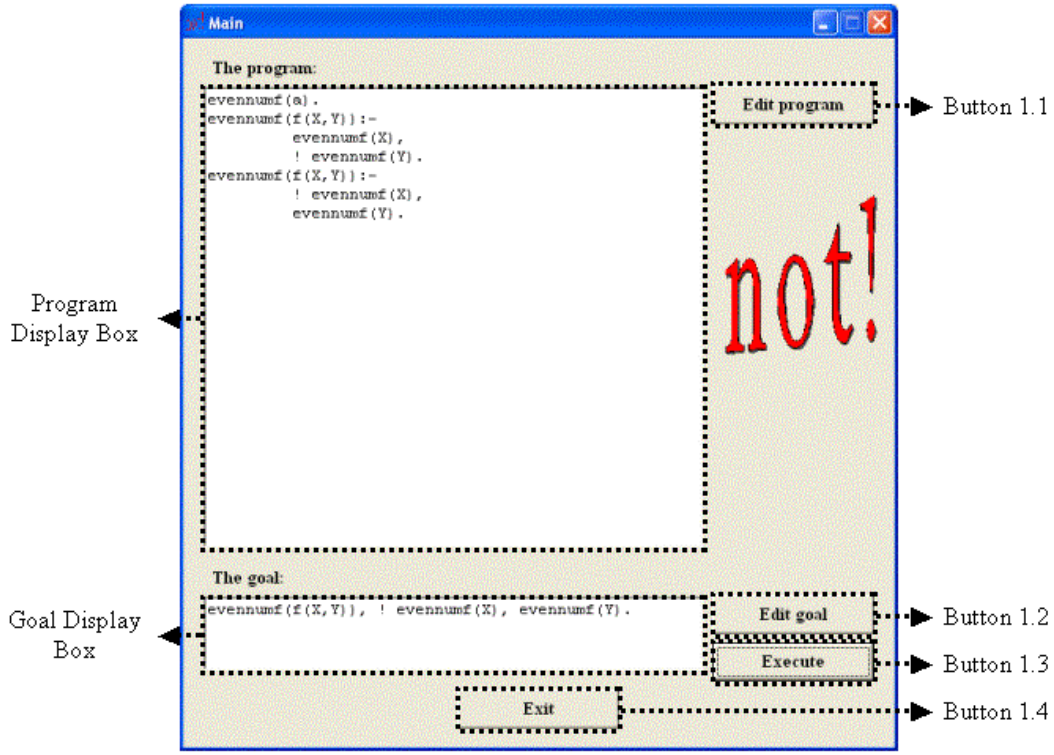
The BCN prototype is started by clicking the file bcn.exe in the installation folder.

Chapter

2

USING THE PROTOTYPE: Main Window

The window that appears when the application is started is the **Main Window**:



BCN Prototype User's Manual. Figure 1: Main Window

From this window, we have access to the whole application:

- the **Button 1.1** allows us to access the **Edit Program Window**,
- the **Button 1.2** allows us to access the **Edit Goal Window**,
- the **Button 1.3** allows us to access the **Execution Window**, and
- the **Button 1.4** allows us to **Exit**.

The **Program Display Box** shows the program that is going to be used during execution.

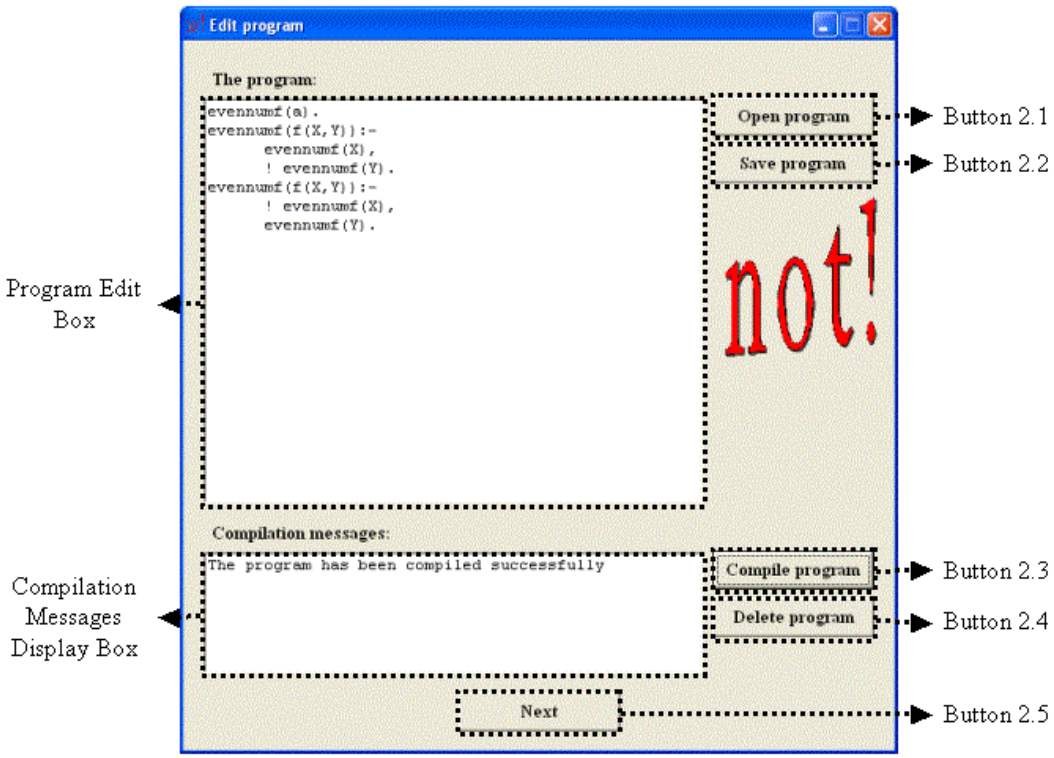
The **Goal Display Box** shows the current goal to be computed.

Chapter

3

WRITING A PROGRAM: Edit Program Window

By a click in the **Button 1.1** of the **Main Window**, the **Edit Program Window** is activated:



BCN Prototype User's Manual. Figure 2: Edit Program Window

Then, a new program can be written in the **Program Edit Box**.

The application deals with normal logic programs. A normal logic program consists in a finite number of clauses of the form $p(\bar{t}(\bar{x})) : -\bar{l}(\bar{t}(\bar{x} \cdot \bar{y}))$. in the usual *Prolog* syntax, except negation that is represented by the symbol '**!**' (Warning: it is not the cut-symbol of *Prolog*).

Warning: do not use names of pre-defined *Prolog* functions.

See the program above.

The functionality of this window is:

- The **Button 2.1** allows us to load (for execution) a program that has been previously saved. Programs are stored in text files with '.pnt' extension.
- The **Button 2.2** allows us to save the program that is currently in the **Program Edit Box**.
- The **Button 2.4** clears the **Program Edit Box**.
- The **Button 2.3** compiles the program that is currently in the **Program Edit Box**. If the program has no errors, a message appears in the **Compilation Messages Display Box**. Otherwise, the errors are shown in this.
- The **Button 2.5** allows us to return to the **Main Window** (in particular, to edit a goal).

Chapter

4

WRITING A GOAL: Edit Goal Window

If a program is already loaded, by clicking **Button 1.2** in the **Main Window**, we can go to the **Edit Goal Window**:

BCN Prototype User's Manual. Figure 3: Edit Goal Window

The goal must be written in the **Goal Edit Box**.

A normal goal is a collection $\bar{l}(\bar{t}(\bar{x}))$. of literals:

- separated by the symbol "**,**",
- the symbol "**!**" is used to represent negation, and
- the symbol "**.**" is used to finish the goal.

See the example in the above window.

The functionality of this window is:

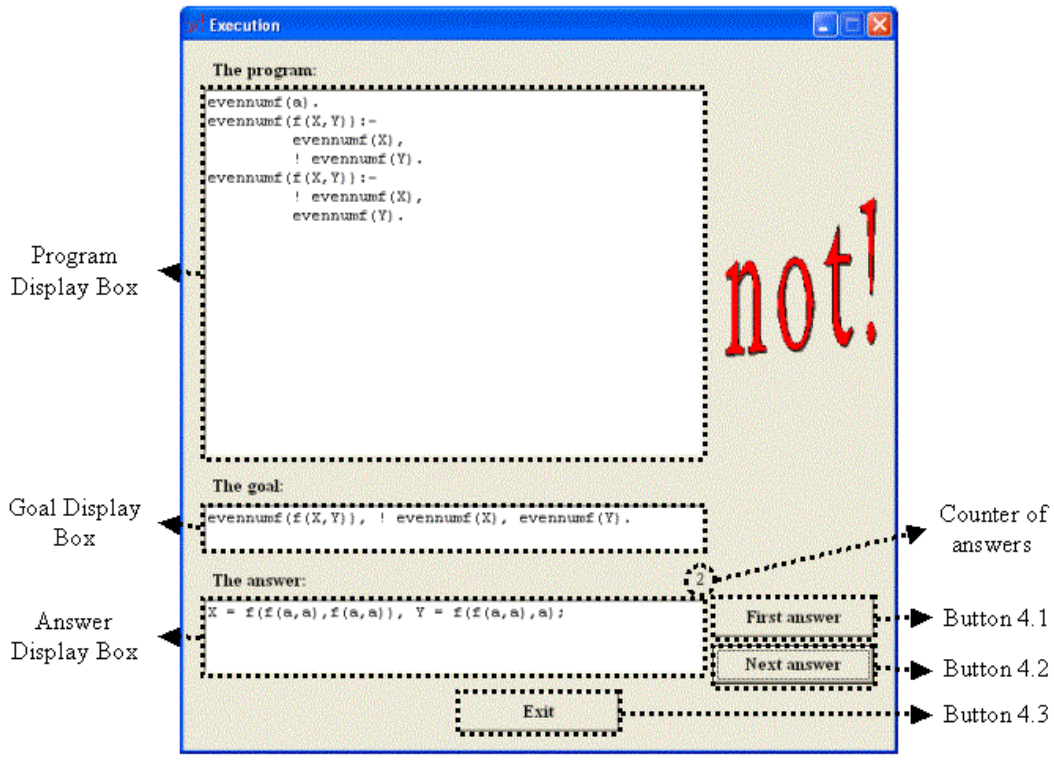
- The **Button 3.1** allows us to load (for computation) a previously saved goal. Goals are stored in text files with '.qnt' extension.
- The **Button 3.2** allows us to save the goal that is currently in the **Goal Edit Box**.
- The **Button 3.4** clears the **Goal Edit Box**.
- The **Button 3.3** starts the compilation of the goal in the **Goal Edit Box**. If it is correct, then a message appears in the **Compilation Messages Display Box**. Otherwise, the errors are shown in this.
- The **Button 3.5** allows us to return to the **Main Window**.

Chapter

5

EXECUTING A PROGRAM: Execution Window

After introducing a program and a goal, the **Button 1.3** starts the computation and opens the **ExecutionWindow**:



BCN Prototype User's Manual. Figure 4: Execution Window

This window is independent of the rest of the application and it is possible to compute at the same time different goals (even, for different programs).

Along the execution, the program and the goal are respectively shown in the **Program Display Box** and the **Goal Display Box**.

The **Answer Display Box** is used to show the successive answers of the given goal and program.

The computation process can be conducted as follows:

- By clicking the **Button 4.1** the computation starts and the first answer (if it exists) is showed in the **Answer Display Box**.
- Then, by clicking the **Button 4.2** the successive answers are computed and shown in the same box.
- When there is not more answers, the message '**`no`**' appears in the **Answer Display Box**.
- At any time, the execution can be restarted by clicking the **Button 4.1**.
- The **Counter of answers** displays the ordinal number of the answer that is currently in the **Answer Display Box**.
- The computation can be finished by clicking the **Button 4.3**.

What is an answer?

- An answer provides information about the variables ($\overline{\mathrm{X}}$) of the goal.

- The general form of an answer is either '**`true`**' or a formula composed by a conjunction of both:
  1. Collapsing equations of the form $\mathrm{X}_i = t(\overline{\mathrm{W}})$, and
  2. Universally quantified collapsing disequations of the form $\forall\overline{\mathrm{V}}(\mathrm{W}_j \neq s(\overline{\mathrm{W}} \cdot \overline{\mathrm{V}}))$, where the term s is not a single variable in $\overline{\mathrm{V}}$ and W_j does not occur in s

  where each X_i occurs at most once.

  Notice that the scope of each universally quantified variable V_i is the disequation where it appears in.

- Hence, an answer involves the variables $\overline{\mathrm{X}}$ of the goal, together with two kind of auxiliary variables:

  - *Prolog*-like variables of the form '`_<char>`' represent the usual existencially quantified variables: these are the variables $\overline{\mathrm{W}}$ in the above notation.
  - Variables of the form '`*<char>`' are used to represent the universally quantified variables: these are the variables $\overline{\mathrm{V}}$ in the above notation.

For example, the answer:

$$\exists W_1 \exists W_2 (X_1 = W_1 \wedge X_2 = W_2 \wedge X_3 = g(W_1) \wedge W_1 \neq a \wedge W_1 \neq W_2 \wedge \forall V(W_1 \neq g(V, W_2)))$$

is displayed in our prototype as follows:

$$X_1 = \_A,\ X_2 = \_B,\ X_3 = g(\_A),\ \_A \neq a,\ \_A \neq \_B,\ \_A \neq f(*C, \_B)$$