A Concurrent Operational Semantics for Constraint Functional Logic Programming

Rafael del Vado Vírseda and Fernando Pérez Morente

TECHNICAL REPORT 03/12
Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
rdelvado@sip.ucm.es fperezmo@fdi.ucm.es

Abstract. In this paper we describe a sound and complete concurrent operational semantics for constraint functional logic programming languages which allows to model declarative applications in which the interaction between demand-driven narrowing and constraint solving helps to prune the search space, leading to shorter goal derivations. We encode concurrency into the generic $CFLP(\mathcal{D})$ scheme, a uniform foundation for the operational semantics of constraint functional logic programming systems parameterized by a constraint solver over the given domain $\mathcal{D}$. In this concurrent version of the $CFLP(\mathcal{D})$ scheme, goal solving processes can be executed concurrently and cooperate together to perform their specific tasks via demand-driven narrowing and declarative residuation guided by constrained definitional trees, constraint solving, and communication by synchronization on logical variables. The paper shows with performance results that the implementation in the $\mathcal{TOY}(\mathcal{FD})$ system of our proposal for the practical instance $CFLP(\mathcal{FD})$ over finite domains $\mathcal{FD}$ can be competitive with respect to other related systems.

1 Introduction

Multiparadigm declarative programming languages and systems [2,6,9,12] aim to integrate the most important declarative programming paradigms, namely *functional programming* (demand-driven rewriting strategies, higher-order facilities, etc.) and *(constraint) logic programming* (goal solving, logical variables, computation with constraints, etc.). The endeavor to extend this declarative combined paradigm to a practical language suitable for concurrent executions has stimulated much research over the last two decades, resulting in a large variety of proposals [3,7,9]. The aim of this research area is the development of *concurrent functional and constraint logic programming* systems [2,9] that maintain the balance between expressiveness and declarative reading: abstraction, computations as proofs, amenability to meta-programming, etc. However, the interactions between all these different features are complex, so the design and implementation of a sound and complete theoretical framework of concurrent and constrained multiparadigm declarative programming systems is non-trivial.

A common feature of the various approaches is the attempt to define declarative operational models for concurrency within the *constraint logic programming scheme* $CLP(\mathcal{D})$ [8], which replaces the basic computational model of logic programming (i.e., *SLD-resolution* with syntactic *unification*) by *constraint solving*

over some *constraint domain* $\mathcal{D}$ (e.g., the integer or the real numbers). The $CLP(\mathcal{D})$ scheme can be generalized into the framework of *concurrent constraint programming* [11] to accommodate a simple and powerful model of declarative concurrent computation based on a *global store*, represented by a constraint on the values that variables can assume. All goal solving processes of the system share this common store, and instead of "reading" and "writing" the values of variables, processes may *ask* (check if a constraint is entailed by the store) and *tell* (augment the store with a new constraint).

The $CFLP(\mathcal{D})$ scheme [10] elegantly captures the fundamental ideas behind the multiparadigm declarative systems, generalizing the $CLP(\mathcal{D})$ scheme to provide uniform foundations for the semantics of functional and constraint logic programming languages. The efficient operational semantics relies on *demand-driven narrowing with definitional trees* [13], a combination of syntactic unification and demand-driven rewriting, parameterized by a constraint solver over the given domain $\mathcal{D}$, which is sound and complete with respect to a declarative semantics formalized by a *constraint rewriting logic* [10], and uses a hierarchical structure called *definitional tree* to efficiently control the computation and maintain the good properties of the *needed narrowing strategy* [7]. The current version of the constraint functional logic system $\mathcal{TOY}$ [12] (and its multiparadigm declarative language) has been designed to efficiently implement the $CFLP(\mathcal{D})$ scheme. However, concurrency is not supported in the $CFLP(\mathcal{D})$ execution model and its efficient implementation in the $\mathcal{TOY}$ system, although declarative forms of concurrency do exist for similar (but less expressive) approaches [3,7,9].

Despite those concurrent extensions, we are not aware of any implementation backed by theoretical results concerning the combination of sound and complete demand-driven narrowing with definitional trees and constraint solving to provide a more powerful declarative integration of concurrent programming techniques. The development of these practical techniques is essential for the implementation of concurrent multiparadigm declarative systems, since they allow further optimizations related to the synchronization and communication of constraint solving mechanisms that can considerably reduce the search space generated by narrowing. For this reason, the motivation of this work is to present a sound and complete operational semantics to describe the concurrent interaction between the computational mechanisms of demand-driven narrowing and constraint solving in the $CFLP(\mathcal{D})$ scheme, providing the theoretical basis of more efficient implementation techniques for the constraint functional logic language $\mathcal{TOY}$ and other related multiparadigm declarative programming systems.

The aim of this paper is to provide a well-founded concurrent operational semantics that has the potential to be at the basis of more efficient implementations of constraint functional logic programming languages than the current ones [4]. Since we focus our work on the operational semantics of the $CFLP(\mathcal{D})$ scheme, the programming language itself does not need new constructs to express explicitly concurrency or to improve its expressivity regarding concurrent features. We center the contribution of this work in the hybrid operational combination between constraint solving and demand-driven narrowing guided by definitional trees, to show that this concurrent operational model allows cons-

traint solving to efficiently reduce the search space generated by narrowing.

The rest of this paper is organized as follows. **Section 2** introduces our approach by presenting an example of declarative concurrency in $CFLP(\mathcal{FD})$, a concrete instance of constraint functional logic programming over the finite domain $\mathcal{FD}$ of integer numbers. In **Sections 3** and **4** we introduce and enrich the presentation of the generic $CFLP(\mathcal{D})$ scheme [10] underlying the implementation of the $\mathcal{TOY}$ system [12], now with concurrent features concerning the combination of demand-driven narrowing and declarative residuation with constrained definitional trees and constraint solving. **Section 5** shows a performance analysis of efficiency and scalability which shows that our concurrent implementation in $\mathcal{TOY}(\mathcal{FD})$ of the concrete instance $CFLP(\mathcal{FD})$ may be competitive with respect to existing multiparadigm declarative constraint programming systems. Finally, **Section 6** summarizes some conclusions and plans for future work.

2 An Example of Concurrent Execution in $CFLP(\mathcal{FD})$

For a first impression of our proposal of a concurrent operational model in constraint functional logic programming, we consider the following *conditional* ($\Leftarrow$) *rewriting rules* ($\rightarrow$) with constraints over the *constraint finite domain* $\mathcal{FD}$ of integers defining a simple $CFLP(\mathcal{FD})$-program to compute *Fibonacci numbers*:

$$\begin{array}{l} \mathit{fib}\,(0) \;\rightarrow 1 \\ \mathit{fib}\,(1) \;\rightarrow 1 \\ \mathit{fib}\,(N) \rightarrow \mathit{fib}\,(N-1) + \mathit{fib}\,(N-2) \Leftarrow N \geq 2 \end{array}$$

In the remaining sections we will use this simple running example to motivate our work and illustrate various technical points. Thus, at this moment the reader can use this example only for a first contact, and later revisit this section to complete the understanding of all the technical details presented in this work. From this program, we want to compute all the values for the variable X from the *user-defined constraint* $\mathit{fib}\,(X) \leq 2$ (i.e., the values 0, 1 and 2 for X). We propose a concurrent operational semantics to improve the efficiency of the sequential $\mathcal{TOY}(\mathcal{FD})$ system [12] implementing $CFLP(\mathcal{FD})$. This enhanced operational semantics begins separating (we use the symbol '$\Box$' for this purpose) the initial goal $\mathit{fib}\,(X) \leq 2$ into the evaluation of a *function call* $\mathit{fib}\,(X) \rightarrow R$ and a solved *constraint store* with a *primitive constraint* $R \leq 2$, introducing a logical variable R for communication and synchronization between both parts:

$$\mathit{fib}\,(X) \rightarrow R \;\Box\; R \leq 2$$

The new system tries to concurrently evaluate both parts: the function call $\mathit{fib}\,(X) \rightarrow R$ by *demand-driven narrowing* [13] (i.e., a combination of *lazy rewriting* '$\rightarrow$' and *unification* '$\mapsto$' by substitutions) for each of the three (variable-renamed) program rules, and the primitive constraint $R \leq 2$ by the $\mathcal{FD}$-*constraint solver* of *SICStus Prolog* underlying $\mathcal{TOY}(\mathcal{FD})$ [12]:

(1) **First program rule:** $\mathit{fib}\,(0) \rightarrow 1$

In order to evaluate the function call $\mathit{fib}\,(X) \rightarrow R$, the first program rule can be applied to instantiate the argument X to 0 (indicated in the goal by

the separation symbol '$\square$' and the unification substitution $\{X \mapsto 0\}$) and to store the corresponding rewriting result 1 in the logical variable R:

$$1 \to R \ \square \ R \leq 2 \ \square \ \{X \mapsto 0\}$$

Now, we can reduce R to 1 and apply the accumulated substitution $\{R \mapsto 1, X \mapsto 0\}$ to instantiate the constraint $R \leq 2$. Then, the $\mathcal{FD}$-constraint solver checks the satisfiability of the instantiated store $1 \leq 2$. Thus, the first answer computed by constrained demand-driven narrowing is $\{X \mapsto 0\}$:

$$\square \ 1 \leq 2 \ \square \ \{R \mapsto 1, X \mapsto 0\} \Rightarrow \boxed{\sigma_1 = \{X \mapsto 0\}} \quad (\textit{First computed answer})$$

(2) **Second program rule:** $\mathit{fib}\,(1) \to 1$

Concurrently, X can be also instantiated to 1 in our computational model, and then the second program rule can be applied to compute the second answer:

$$1 \to R \ \square \ R \leq 2 \ \square \ \{X \mapsto 1\}$$
$$\square \ 1 \leq 2 \ \square \ \{R \mapsto 1, X \mapsto 1\} \Rightarrow \boxed{\sigma_2 = \{X \mapsto 1\}} \quad (\textit{Second computed answer})$$

(3) **Third program rule:** $\mathit{fib}\,(X) \to \mathit{fib}\,(X-1) + \mathit{fib}\,(X-2) \Leftarrow X \geq 2$

Also concurrently, a variable-renamed of the third program rule can be applied to the goal, resulting in the evaluation of two new *fib* function calls:

$$\mathit{fib}\,(X-1) + \mathit{fib}\,(X-2) \to R \ \square \ X \geq 2,\ R \leq 2$$
$$\mathit{fib}\,(X-1) \to R_1,\ \mathit{fib}\,(X-2) \to R_2 \ \square \ X \geq 2,\ R_1 + R_2 = R,\ R \leq 2$$

In this third case (3), our enhanced version of the $\mathcal{TOY}(\mathcal{FD})$ system explores concurrently two possible ways to efficiently compute more answers, according to the two possible flows of communication and synchronization (i.e., instantiation of the common logical variables X, R_1, R_2 and R) between the mechanisms of demand-driven narrowing and constraint solving, differing in their length (and therefore in efficiency) due to the different concurrent interleavings of both computational mechanisms.

(3.1) **From narrowing to constraint solving:** In this first case, closer to the sequential execution of the $\mathcal{TOY}(\mathcal{FD})$ system [12], our operational model evaluates concurrently the function calls $\mathit{fib}\,(X-1)$ and $\mathit{fib}\,(X-2)$ (or equivalently, the flattened and standardized forms $\mathit{fib}\,(N_1)$ and $\mathit{fib}\,(N_2)$ with new constraints $N_1 = X - 1$ and $N_2 = X - 2$, respectively, in the common constraint store) by applying again a combination of demand-driven narrowing and constraint solving. For example, the system can compute the value $\{X \mapsto 2\}$ for X applying concurrently the second and the first program rules, respectively, to compute $\{N_1 \mapsto 1\}$ and $\{N_2 \mapsto 0\}$, and then applying the constraint solver to $1 = X - 1$ and $0 = X - 2$. Then, the corresponding

result 1 will be stored in the logical variables R_1 and R_2:

$1 \rightarrow R_1, 1 \rightarrow R_2 \,\square\, X \geq 2, R_1 + R_2 = R, R \leq 2, 1 = X-1, 0 = X-2 \,\square\, \{N_1 \mapsto 1, N_2 \mapsto 0\}$
$1 \rightarrow R_1, 1 \rightarrow R_2 \,\square\, R_1 + R_2 = R,\ R \leq 2 \,\square\, \{N_1 \mapsto 1, N_2 \mapsto 0\, X \mapsto 2\,\}$

To ensure the consistency of this evaluation process by the demand-driven narrowing computation, our concurrent operational model has to protect (or *suspend*) the evaluation of variables R_1 and R_2 from the action of the constraint solver in favour of an evaluation only by narrowing to compute $\{R_1 \mapsto 1, R_2 \mapsto 1\}$ from $1 \rightarrow R_1$ and $1 \rightarrow R_2$. Analogously, since we want to compute values $\{N_1 \mapsto 1, N_2 \mapsto 0\}$ for the variables N_1 and N_2 by narrowing, we also need to protect both variables from the constraint solver action (this is the so-called *flex* narrowing option in this work). Finally, since both processes are synchronized by sharing the common constraint store that contains $R_1 + R_2 = R$, $R \leq 2$, and we have computed by narrowing the values $\{R_1 \mapsto 1, R_2 \mapsto 1\}$, the constraint solver can compute now the substitution $\{R \mapsto 2\}$ and offer to the user the third computed answer $\{X \mapsto 2\}$:

$\square\, 1 + 1 = R,\ R \leq 2 \,\square\, \{N_1 \mapsto 1, N_2 \mapsto 0, X \mapsto 2, R_1 \mapsto 1, R_2 \mapsto 1\}$
$\square\, 2 \leq 2 \,\square\, \{N_1 \mapsto 1, N_2 \mapsto 0, X \mapsto 2, R_1 \mapsto 1, R_2 \mapsto 1, R \mapsto 2\} \Rightarrow \boxed{\sigma_3 = \{X \mapsto 2\}}$

At this point, the narrowing computation in $\mathcal{TOY}(\mathcal{FD})$ performs an infinite and useless "trial and error" generation of other possible values for X to find new possible answers. For example, alternatively applying the first and second program rules it is possible to compute other values for N_1 and N_2 due to the concurrent evaluation of *fib* (N_1) and *fib* (N_2): $\{N_1 \mapsto 0, N_2 \mapsto 0\}$, $\{N_1 \mapsto 0, N_2 \mapsto 1\}$ or $\{N_1 \mapsto 1, N_2 \mapsto 1\}$. All of these concurrent processes only obtain inconsistent values for X from $N_1 = X - 1$ and $N_2 = X - 2$ and must be discarded. Moreover, for each application of the third program rule, we have to evaluate two new function calls *fib* in order to infinitely compute concrete values for R_1 and R_2, and to check that each concrete instance of the constraint store $R_1 + R_2 = R$, $R \leq 2$ fails. How can our concurrent operational model efficiently helps to prevent this infinite and useless search space generated by narrowing? This is our main paper's idea:

(3.2) **From constraint solving to narrowing:** In this case, our concurrent operational model needs to protect (or *suspend*) variables N_1 and N_2, now from the narrowing action (this is the so-called *rigid* narrowing or *residuation* option in this work). Then, as an important difference with respect to (3.1), the solver allows to solve the constraint $X \geq 2$ to generate and assign directly to the variables X, $N_1 = X - 1$ and $N_2 = X - 2$ only correct integer values: $\{X \mapsto 2, N_1 \mapsto 1, N_2 \mapsto 0\}$, $\{X \mapsto 3, N_1 \mapsto 2, N_2 \mapsto 1\}$, $\{X \mapsto 4, N_1 \mapsto 3, N_2 \mapsto 2\}$, etc. For each of these possible values, the system creates a process and awakes simple concurrent applications of *rewriting* (instead of expensive "trial and error" applications of narrowing as we have seen in (3.1)). For example, for the values $\{X \mapsto 2, N_1 \mapsto 1, N_2 \mapsto 0\}$ the concurrent system computes the same third answer $\{X \mapsto 2\}$ in less time. For any other value $X \geq 3$, this process is free to use, concurrently, efficient

$\mathcal{FD}$-constraint solving techniques [1,4] to add directly to the constraint store $R_1+R_2 > 2$. Then, the solver fails checking the extended common constraint store $R_1 + R_2 > 2$, $R_1 + R_2 = R$, $R \leq 2$ and stops the generation of more values for X. Moreover, since the goal solving processes share the same constraint store, the (3.2) option kills automatically all the remaining active processes in the (3.1) option, avoiding the generation of an infinite and useless narrowing computation. In conclusion, in this case constraint solving has helped to efficiently compute the last answer, and at the same time has reduced the search space generated by narrowing.

In the following sections we have formalized, implemented and applied all these techniques to other classical examples and benchmarks in constraint functional logic programming over finite domains [4] including other interesting features of the $CFLP(\mathcal{FD})$-programming (e.g., the use of infinite data structures and higher-order programming facilities) to confirm how this concurrent interaction between constraint solving and narrowing can prune the search space generated by $\mathcal{TOY}(\mathcal{FD})$. In the next two sections, we formalize the logical aspects of this operational model to obtain a novel sound and complete hybrid execution model of declarative constraint concurrency.

3 Concurrent Constraint Functional Logic Programming

In this section we give a revised summary of the generic $CFLP(\mathcal{D})$ scheme [10] underlying our proposal of a concurrent system for multiparadigm declarative constraint programming. We need to extend and enrich this programming scheme from its theoretical foundations in [10] to integrate sound and complete concurrent features concerning the combination of demand-driven narrowing and declarative residuation with constrained definitional trees and constraint solving.

3.1 Expressions, Patterns, and Constraints

A *signature* is a tuple $\Sigma = \langle DC, FS \rangle$ where $DC = \bigcup_{n\in\mathbb{N}} DC^n$ and $FS = \bigcup_{n\in\mathbb{N}} FS^n$ are families of countably infinite and mutually disjoint sets of *data constructors* and *evaluable function symbols*. Evaluable functions can be further classified into domain dependent *primitive functions* $PF^n \subseteq FS^n$ (e.g. $+, \leq \in PF^2$) and user *defined functions* $DF^n = FS^n \setminus PF^n$ for each arity $n \in \mathbb{N}$ (e.g. $fib \in DF^1$). We also assume a countably infinite set Var of *variables* $X, Y, \ldots$ and a set $\mathcal{U}$ of *primitive elements* $u, v, \ldots$ (as e.g. the set $\mathbb{Z}$ of integer numbers).

Expressions $e, e' \in Exp(\mathcal{U})$ have the syntax $e ::= \bot \mid u \mid X \mid h \mid (e\, e')$, where $\bot$ is a special symbol in DC^0 to denote an undefined data value, $u \in \mathcal{U}$, $X \in Var$, and $h \in DC \cup FS$. The following classification of expressions is useful: $X\,\overline{e}_m$ with $X \in Var$ and $m \geq 0$ is called a *flexible expression*, while $u \in \mathcal{U}$ and $h\,\overline{e}_m$ with $h \in DC \cup FS$ are called *rigid expressions*. Moreover, a rigid expression $h\,\overline{e}_m$ is called *active* if and only if $h \in FS^n$ and $m \geq n$, and *passive* otherwise. The occurrence of a symbol is *passive* if and only if is a primite element $u \in \mathcal{U}$ or is the root symbol h of a passive expression (a symbol used in this sense is called a *passive symbol*). Another class of expressions are *Patterns* $s, t \in Pat(\mathcal{U})$, built as $t ::= \bot \mid u \mid X \mid c\,\overline{t}_m \mid f\,\overline{t}_m$, where $c \in DC^n$ $(m \leq n)$ and $f \in FS^n$ $(m < n)$.

For every expression e, the set of *positions* in e is inductively defined as follows: the empty sequence identifies e itself, and for every expression of the form $h\overline{e}_m$, the sequence $i \cdot q$, where i is a positive integer not greater than m and q is a position, identifies the subexpression of e_i at q. The subexpression of e at p is denoted by $e|_p$ and the result of *replacing* $e|_p$ with e' in e is denoted by $e[e']_p$. If e is a *linear* expression (without repeated variable occurrences), $pos(X, e)$ will be used for the position of the variable X occurring in e. *Substitutions* $\sigma \in Sub(\mathcal{U})$ are mappings $\sigma : \mathcal{V} \to Pat(\mathcal{U})$ extended homomorphically to $\sigma : Exp(\mathcal{U}) \to Exp(\mathcal{U})$. We define the *domain* $Dom(\sigma)$ of a substitution σ as the collection of variables that are not mapped to themselves.

A *constraint domain* $\mathcal{D}$ provides a set of specific data elements $u \in \mathcal{U}$ along with certain primitive functions $p \in PF$ operating on them. For example, the *constraint finite domain* $\mathcal{FD}$ [4,10] can be formalized as a structure with carrier set consisting of patterns built from the symbols in a signature Σ and the set of primitive elements $\mathbb{Z}$. Symbols in Σ are intended to represent data constructors (e.g., the list constructors), domain specific primitive functions (e.g., addition and multiplication over $\mathbb{Z}$), and user defined functions. *Constraints* have the syntactic form $p\,\overline{e}_n$, with $p \in PF^n$ a primitive relational symbol and $\overline{e}_n \in Exp(\mathcal{U})$ (e.g. $fib\,(X) \leq 2$, $X \geq 2$ or $R_1 + R_2 = R$ in infix notation).

3.2 Programs and Constrained Definitional Trees

In the sequel, we assume an arbitrarily fixed constraint domain $\mathcal{D}$ built over a set of primitive elements $\mathcal{U}$. In this setting, a *program* is a set of constrained rewrite rules that defines the behavior of possibly higher-order and/or non-deterministic lazy functions over $\mathcal{D}$, called *program rules*. More precisely, a program rule R for $f \in DF^n$ has the form $f\,\overline{t}_n \to r \Leftarrow P \,\square\, C$ (abbreviated as $f\,\overline{t}_n \to r$ if P and C are both empty; see **Section 2**) and is required to satisfy:

- The left-hand side $f\,\overline{t}_n$ (e.g. $fib(N)$) is a linear expression with $\overline{t}_n \in Pat(\mathcal{U})$ and the right-hand side $r \in Exp(\mathcal{U})$.
- P is a finite sequence of so-called *productions* of the form $e_i \to R_i$ $(1 \leq i \leq k)$, intended to be interpreted as a conjunction of local definitions with no cycles [10] (e.g. $fib(X - 1) \to R_1, fib(X - 2) \to R_2$). For all $1 \leq i \leq k$, $e_i \in Exp(\mathcal{U})$, and $R_i \notin Var(f\,\overline{t}_n)$ are different variables.
- C is a finite set of constraints (e.g. $X \geq 2, R_1 + R_2 = R, R \leq 3$), also intended to be interpreted as a conjunction, and possibly including occurrences of user-defined function symbols.

$\mathcal{T}_\tau$ is a *constrained Definitional Tree* over $\mathcal{D}$ ($cDT(\mathcal{D})$ for short) with *call pattern* τ (a linear pattern of the form $f\overline{t}_n$, where $f \in DF^n$ and $\overline{t}_n \in Pat(\mathcal{U})$) if its depth is finite and one of the following cases holds for the rules of a program $\mathcal{P}$:

- $\mathcal{T}_\tau \equiv \underline{rule}(\tau \to r_1 \Leftarrow P_1 \,\square\, C_1 \,\|\, \ldots \,\|\, r_m \Leftarrow P_m \,\square\, C_m)$, where $\tau \to r_i \Leftarrow P_i \,\square\, C_i$ for all $1 \leq i \leq m$ are variants of overlapping program rules in $\mathcal{P}$.
- $\mathcal{T}_\tau \equiv \underline{case}(\tau, X, op, [\mathcal{T}_1, \ldots, \mathcal{T}_k])$, where X is a variable in τ, $op \in \{flex, rigid, flex/rigid\}$, $h_1, \ldots, h_k$ $(k > 0)$ are pairwise different passive symbols of $\mathcal{P}$, and for all $1 \leq i \leq k$, $\mathcal{T}_i$ is a $cDT(\mathcal{D})$ with call pattern $\tau\sigma_i$, where $\sigma_i = \{X \mapsto h_i\overline{Y}_{m_i}\}$ with $\overline{Y}_{m_i}$ new distinct variables such that $h_i\overline{Y}_{m_i} \in Pat(\mathcal{U})$.

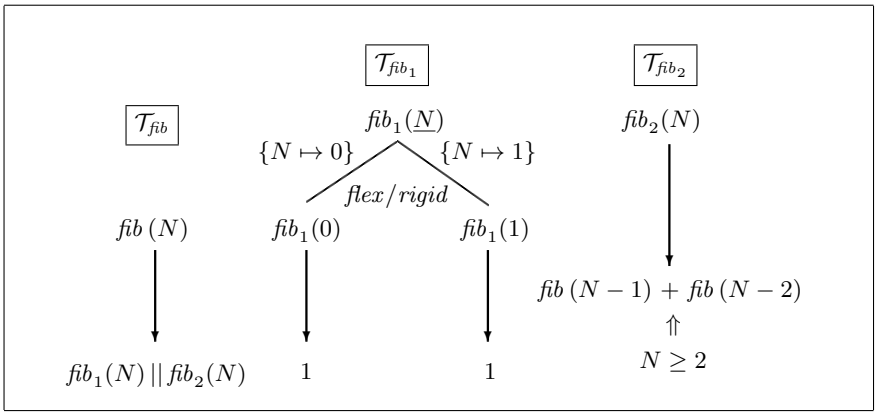

Fig. 1. Constrained definitional trees for declarative concurrency in $CFLP(\mathcal{FD})$.

A $\mathcal{T}_f$ *of a function symbol* $f \in DF^n$ defined by a program $\mathcal{P}$ is a $cDT(\mathcal{D})$ with call pattern $f\overline{X}_n$, where $\overline{X}_n$ are new variables, and the collection of all the program rules obtained from the different $\underline{rule}$ nodes equals, up to variants, the collection of all the program rules defining f in $\mathcal{P}$. As an example, a graphical representation of a $cDT(\mathcal{FD})$ for the *fib* function defined in **Section 2**

$$\mathcal{T}_{fib} \equiv \underline{rule}\,(fib\,(N) \;\rightarrow\; fib_1(N) \;||\; fib_2(N))$$
$$\mathcal{T}_{fib_1} \equiv \underline{case}\,(fib_1(N), N, flex/rigid, [\,\underline{rule}\,(fib_1\,(0) \rightarrow 1), \underline{rule}\,(fib_1\,(1) \rightarrow 1)])$$
$$\mathcal{T}_{fib_2} \equiv \underline{rule}\,(fib_2(N) \;\rightarrow\; fib\,(N-1) + fib\,(N-2) \;\Leftarrow\; N \geq 2)$$

is given in **Fig. 1.** As a novelty with respect to other related works on definitional trees [6,7,13], our operational model can use now the special option *flex/rigid* to initiate the concurrent evaluation of both alternatives, *flex*-narrowing option and *rigid*-narrowing/residuation option (as we have seen in cases (3.1) and (3.2) in **Section 2**). The pre-runtime construction of constrained definitional trees for the functions defined in a program follows the algorithmic ideas explained in [12] and **Appendix**, displaying now an extra *flag* to simultaneously initiate and identify each of the two possible concurrent options *flex* and *rigid*.

3.3 Goals and Answers

A *goal* G for a program has the general form $P \,\square\, C \,\square\, S \,\square\, \sigma$, where the separation symbol '$\square$' must be interpreted as a conjunction, and:

- $P \equiv e_1 \rightarrow R_1, \ldots, e_n \rightarrow R_n$ is a finite conjunction of so-called *productions*, where each R_i is a distinct variable and e_i is an expression (we call these productions as *suspensions*), or a pair of the form $< \tau, \mathcal{T} >$ with τ an instance of the call pattern in the root of a $cDT(\mathcal{D})$ $\mathcal{T}$ (we call these productions as *demanded productions*). The set of *produced variables* is $PVar(P) =_{def} \{R_1, \ldots, R_n\}$ (e.g. R, R_1 and R_2 in **Section 2**).

- $C \equiv \delta_1, \ldots, \delta_k$ is a finite conjunction of constraints (possibly including user-defined function symbols; e.g. $fib\,(X) \leq 2$ in the initial goal of **Section 2**).
- $S \equiv \pi_1, \ldots, \pi_l$ is a finite conjunction of *primitive* constraints (i.e., constraints with only pattern arguments; e.g. $R \leq 2$), called *constraint store*.
- $\sigma \in Sub(\mathcal{U})$ is an idempotent substitution called *answer substitution* such that $Dom(\sigma) \cap Var(P \,\square\, C \,\square\, S) = \emptyset$. For example, $\sigma_3 = \{X \mapsto 2\}$ in **Section 2**.

A *solved goal* is a goal $\square \,\square\, S \,\square\, \sigma$ in which P and C are empty, and identifies an *answer* $S \,\square\, \sigma$ (or simply σ, as we have seen in **Section 2**). We say that $X \in Var(G)$ is a *demanded variable* in G if and only if one of the following cases holds:

(1) Any substitution that is a solution of S cannot bind X to the undefined value $\perp$ (shortly, $X \in DVar_{\mathcal{D}}(S)$). For example, $R \in DVar_{\mathcal{FD}}(R \leq 3)$.

(2) There exists a suspension $(X\overline{a}_k \rightarrow R) \in P$ such that $k > 0$ and R is a demanded variable in G (this case is only necessary to deal with higher-order [10]).

(3) There exists a demanded production $(< e, \underline{case}\,(\tau, Y, op, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \rightarrow R) \in P$ such that $X = e|_{pos(Y,\tau)}$ and R is a demanded variable in G (see e.g. N_1 and N_2 in (3.1) and (3.2)). If op in the branch node is of type *flex*, the variable X is called a *flex variable* (e.g. N_1 and N_2 in (3.1)). Otherwise, the variable X is called a *rigid variable* (e.g. N_1 and N_2 in (3.2)).

We write $DVar_{\mathcal{D}}(G)$ (or more precisely $DVar_{\mathcal{D}}(P \,\square\, S)$ or $DVar_{\mathcal{D}}(S)$) for the *set of demanded variables* in the goal G, and $FVar(G)$, $RVar(G)$ (or more precisely $FVar(P)$, $RVar(P)$) for the *set of flex and rigid variables*, respectively.

4 A Concurrent Operational Semantics for $CFLP(\mathcal{D})$

In this section we present a set of *concurrent goal transformation rules* of the form $G \Vdash_{\mathbf{R}} \|_{i=1}^{k} G_i$, specifying all the possible *concurrent evaluations* ($\|_{i=1}^{k}$) of subgoals G_i obtained by applying a rule $\mathbf{R}$ of goal solving ($\Vdash_{\mathbf{R}}$) to a goal G in our concurrent operational semantics for the $CFLP(\mathcal{D})$ scheme. As we will see in **Section 5**, all these rules (formally presented in **Figs. 2** and **3** and **Appendix**) have been implemented in the $\mathcal{TOY}$ system [12] and are implicitly applied in our running example of **Section 2**. We refer the reader to that section and [14] for detailed examples illustrating the application of all these rules. We write $G \Vdash^* \|_{i=1}^{k} G_i$ to represent *concurrent derivations*, given by the successive application ($\Vdash^*$) of concurrent goal transformation rules from G. For example, the concurrent derivation $G \Vdash^* G_1' \,\|\, G_{21} \,\|\, G_{22}$ represents the concurrent goal transformation steps $G \Vdash G_1 \,\|\, G_2$ with $G_1 \Vdash G_1'$ and $G_2 \Vdash G_{21} \,\|\, G_{22}$.

Each time a goal G contains the conjunction of two or more atomic statements that could be concurrently evaluated (e.g., two or more productions), our operational model creates concurrent goal solving processes, each of one consisting of an atomic statement from G, together with the necessary information for

an adequate and consistent demand-driven evaluation applying a concurrent goal transformation rule (i.e., the sets of produced, demanded, rigid and flex variables). Moreover, for synchronization and in order to properly combine the possible computed answers from subgoal processes, as well as the cases in which processes remain *suspended* (indicated by the symbol $\circlearrowleft$) or *fail* (indicated by the symbol ■), new subgoals must share the constraint store of the main goal G (similarly to the *concurrent constraint programming* approach [11] for the $CLP(\mathcal{D})$ scheme described in **Section 1**).

4.1 Concurrent Demand-Driven Narrowing and Residuation

The goal transformation rules concerning concurrent and constrained demand-driven narrowing with constrained definitional trees (see **Fig. 2.**) allow us to combine the most important operational principles of concurrent and constrained multiparadigm declarative programming [6]: the demand-driven narrowing strategy for functional logic languages [13], the problem-solving capabilities of constraint logic programming [8], and the residuation principle with constrained definitional trees [7] to add the possibility of efficient concurrent computations.

We start with a suspension $e \to R$ representing the computation of a function call, for example $\mathit{fib}\,(X) \to R$ (other rules for suspensions can be found in **Appendix** and can be easily adapted from earlier works [4] on the sequential $CFLP(\mathcal{D})$ scheme), where e has a user-defined function symbol f in the root (e.g. *fib*) and R is a demanded variable (e.g. by the constraint store $R \leq 2$). Then, the rule **DT** is applicable, awakening the suspension $e \to R$, decorating e with an appropriate $cDT(\mathcal{D})$ $\mathcal{T}_f$ (e.g. $\mathcal{T}_{\mathit{fib}}$ in **Fig. 1**), and introducing a new demanded production $< e, \mathcal{T}_f > \to R$ into the goal. If the function call is not demanded (i.e., R is not a demanded variable), this computation remains suspended ($\circlearrowleft$) until the variable R disappears from the goal (and then the suspension can be eliminated) or R becomes demanded. The goal transformation rules for demanded productions $< e, \mathcal{T}_f > \to R$ encode the *demand-driven narrowing strategy* [7,13] guided by the constrained definitional tree $\mathcal{T}_f$, now in a concurrent setting:

- If $\mathcal{T}_f$ is a *rule* tree, then the transformation **RRA** can be concurrently applied ($\|_{i=1}^{k}$) for each of the k available overlapping program rules for rewriting e, introducing appropriate suspensions and constraints into the new subgoals so that a demand-driven evaluation can be ensured.
- If $\mathcal{T}_f$ is a *case* tree, one of the transformations **CSS**, **CC**, **DN**, **DP**, **DR** or **DI** must be applied, according to the kind of symbol h occurring in e at the case-distinction position $\mathit{pos}(X, \tau)$:
  - If h is a passive symbol h_i, then **CSS** selects the appropriate subtree $\mathcal{T}_i$ (otherwise **CC** fails ■).
  - If h is an active primitive or defined function symbol g, then **DN** introduces a new demanded suspension in the goal to evaluate $e|_{\mathit{pos}(X,\tau)}$.
  - If h is a produced variable Y, the goal must remain suspended ($\circlearrowleft$) using **DP** until a concurrent process of the computation evaluates Y.

DT Definitional Tree

$$f\overline{e}_n \to R, P \,\square\, C \,\square\, S \,\square\, \sigma \Vdash_{\mathbf{DT}} \left\{ \begin{array}{ll} < f\overline{e}_n, \mathcal{T}_{f\overline{X}_n} > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma & \text{if } R \in DVar_{\mathcal{D}}(P \,\square\, S) \\ \circlearrowleft & \text{if } R \notin DVar_{\mathcal{D}}(P \,\square\, S) \end{array} \right\}$$

if $f \in DF^n$, and all variables in $\mathcal{T}_{f\overline{X}_n}$ are new variables.

RRA Rewrite Rule Application

$$< f\overline{e}_n, \underline{rule}\,(f\overline{t}_n \to r_1 \Leftarrow P_1 \,\square\, C_1 \parallel \ldots \parallel r_k \Leftarrow P_k \,\square\, C_k) > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma \Vdash_{\mathbf{RRA}} \|_{i=1}^{k}\ r_i \to R, P_i, P \,\square\, \overline{e_n = t_n}, C_i, C \,\square\, S \,\square\, \sigma$$

CSS Case Selection

$$< e, \underline{case}\,(\tau, X, op, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma \Vdash_{\mathbf{CSS}} < e, \mathcal{T}_i > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma$$

if $e|_{pos(X,\tau)} = h_i \ldots$ with $1 \leq i \leq k$ given by e, and h_i is the passive symbol associated to $\mathcal{T}_i$.

CC Case non-Cover

$$< e, \underline{case}\,(\tau, X, op, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma \Vdash_{\mathbf{CC}} \blacksquare$$

if $e|_{pos(X,\tau)} = h \ldots$ is a passive symbol but $h \notin \{h_1, \ldots, h_k\}$.

DN Demand Narrowing

$$< e, \underline{case}\,(\tau, X, op, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma \Vdash_{\mathbf{DN}} e|_{pos(X,\tau)} \to R', < e[R']_{pos(X,\tau)}, \underline{case}\,(\tau, X, op, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma$$

if $e|_{pos(X,\tau)} = g \ldots$ with $g \in FS$ active (primitive or defined function), and R' new variable.

DP Demand Produced Variable

$$< e, \underline{case}\,(\tau, X, op, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma \Vdash_{\mathbf{DP}} \circlearrowleft$$

if $e|_{pos(X,\tau)} = Y$ with $Y \in PVar(P)$.

DR Demand Residuation

$$< e, \underline{case}\,(\tau, X, \mathit{rigid}, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma \Vdash_{\mathbf{DR}} \circlearrowleft$$

if $e|_{pos(X,\tau)} = Y$ with $Y \notin PVar(P)$.

DI Demand Instantiation

$$< e, \underline{case}\,(\tau, X, \mathit{flex}, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \to R, P \,\square\, C \,\square\, S \,\square\, \sigma \Vdash_{\mathbf{DI}} \|_{i=1}^{k}\ (< e, \mathcal{T}_i > \to R, P \,\square\, C \,\square\, S)\sigma_i \,\square\, \sigma\sigma_i$$

if $e|_{pos(X,\tau)} = Y$ with $Y \notin PVar(P)$, and $\sigma_i = \{Y \mapsto h_i \overline{Y}_{m_i}\}$ with h_i $(1 \leq i \leq k)$ the passive symbol associated to $\mathcal{T}_i$, and $\overline{Y}_{m_i}$ are new variables.

Fig. 2. Rules for concurrency in constrained demand-driven narrowing and residuation.

AC Atomic Constraint

$$P \square\ p\overline{e}_n, C \square S \square \sigma \Vdash_{\mathbf{AC}} \|_{i=1}^{n}\ e_i \to X_i, P \square C \square\ p\overline{X}_n, S \square \sigma$$

if $p \in PF^n$, $p\overline{e}_n$ is a constraint, and $\overline{X}_n$ are new variables.

CS Constraint Solving

$$P \square C \square S \square \sigma \Vdash_{\mathbf{CS}\{\chi\}} \|_{i=1}^{k} \left\{ \begin{array}{ll} (P \square C)\sigma_i \square S_i \square \sigma\sigma_i & \text{if (1) or (2) or (3)} \\ \circlearrowleft & \text{otherwise} \end{array} \right\}$$

if $Solver^{\mathcal{D}}(S, \chi) = \bigvee_{i=1}^{k}(S_i \square \sigma_i)$ with $\chi =_{def} PVar(P) \cup FVar(P)$.

SF Solving Failure

$$P \square C \square S \square \sigma \Vdash_{\mathbf{SF}\{\chi\}} \blacksquare \qquad \text{if } Solver^{\mathcal{D}}(S, \chi) = \mathit{fail}.$$

Fig. 3. Rules for concurrent constraint solving.

- If Y is a non-produced variable, there are two possibilities:
  - If the branch node has the option *rigid* (or *flex/rigid*), we must suspend the evaluation ($\circlearrowleft$) using **DR** until the variable has been bound, for example, by the action of the constraint solver (as we have seen in (3.2) for N_1 and N_2, and we will formalize in the next subsection). This case corresponds to the computational principle of declarative *residuation* [7].
  - If the branch node has the option *flex* (or *flex/rigid*), then **DI** selects concurrently ($\|_{i=1}^{k}$) each subtree $\mathcal{T}_i$, generating an appropriate binding σ_i for Y (as e.g. for N_1 and N_2 in (3.1)).

4.2 Concurrent Constraint Solving

The goal transformation rules concerning *concurrent constraint solving* (see **Fig. 3.**) are designed to concurrently combine the evaluation of (primitive or user-defined) constraints with the action of a constraint solver over the given domain. The first rule **AC** evaluates non-primitive constraints $p\overline{e}_n$ (e.g. $\mathit{fib}\,(X) \leq 2$) by performing a concurrent evaluation ($\|_{i=1}^{n}$) of their arguments e_i in suspensions $e_i \to X_i$, and introducing a flattened primitive constraint $p\overline{X}_n$ into the common constraint store, with new logical variables $\overline{X}_n$ for the communication and synchronization among all these concurrent goal solving processes.

For the evaluation of primitive constraints in a constraint domain $\mathcal{D}$ we postulate a *constraint solver* of the form $Solver^{\mathcal{D}}(S, \chi)$, which can reduce any given finite conjunction of primitive constraints S representing the constraint store of

the goal into an equivalent simpler solved form. The constraint solver needs to take proper care of a selected set of so-called *critical* (or *protected*) *variables* $\chi =_{def} PVar(P) \cup FVar(P)$ occurring in S to ensure a correct demand-driven evaluation (see variables R_1, R_2 and N_1, N_2 in (3.1) and (3.2) of **Section 2**). We require that any solver invocation returns a finite disjunction of k simpler solved form alternatives $S_i \,\square\, \sigma_i$. Then, the rule **CS** describes the possible concurrent ($\|_{i=1}^{k}$) evaluations of a single goal by a solver's invocation for each possible alternative solved form computed by the constraint solver. For example, in (3.2):

$$\begin{aligned} Solver^{\mathcal{FD}}(X \geq 2, R_1 + R_2 = R, R \leq 2, N_1 = X - 1, N_2 = X - 2, \{R_1, R_2\}) = \\ (R_1 + R_2 = R, R \leq 2 \,\square\, \{X \mapsto 2, N_1 \mapsto 1, N_2 \mapsto 0\}) \vee \\ (X \geq 3, R_1 + R_2 > 2, R_1 + R_2 = R, R \leq 2, N_1 = X - 1, N_2 = X - 2 \,\square\, \{\,\}) \end{aligned}$$

To avoid deadlock situations, and to guarantee the formal properties of completeness and consistency for the concurrent demand-driven evaluation strategy, we require solvers to have the ability to compute and discriminate a distinction of the following cases and situations for each concurrent solved form alternative (illustrated by cases (3.1) and (3.2) in **Section 2**):

(1) a suspended production (↺) (for example, suspended by the **DT** rule) with a non-demanded critical variable at the right-hand side may be now demanded (and then activated) by the new constraint store S_i of some alternative $S_i \,\square\, \sigma_i$ (formally, $DVar_{\mathcal{D}}(S_i) \cap \chi \neq \emptyset$), or

(2) a suspended demanded production (↺) (for example, suspended by the **DR** rule) could be activated by applying σ_i to instantiate a *rigid* and not produced variable in this production (i.e., $Dom(\sigma_i) \cap RVar(P) \neq \emptyset$), or

(3) a suspended production (↺) could be irrelevant for the new constraint store S_i (i.e., $Var(S_i) \cap \chi = \emptyset$) and then has to be eliminated. This condition is crucial to perform a demand-driven evaluation and to avoid unnecessary computations in the search space generated by narrowing.

For example, the first alternative computed by $Solver^{\mathcal{FD}}$ in (3.2) corresponds to the situation (2) because we have an instantiation of the rigid variables N_1 and N_2. For any other situation, the corresponding goal solving process must be suspended (↺) by the action of the constraint solver. Additionally, the failure rule **SF** is used for failure detection (■) in the constraint solving process. For example, the second alternative computed by $Solver^{\mathcal{FD}}$ in (3.2) corresponds to solving failure because we have an inconsistent store that contains $R_1 + R_2 > 2, R_1 + R_2 = R, R \leq 2$. Therefore, the rule **SF** can be applied.

A complete and detailed example of a concurrent derivation, using the set of transformation rules in **Figs. 2** and **3** and **Appendix**, and corresponding to the explanations given in **Section 2** for our running example, can be found in [14]. Moreover, the reader can find in [14] other examples of concurrent goal solving which highlight the main properties of our concurrent operational model in constraint functional logic programming over finite domains.

4.3 Soundness and Completeness

We conclude this section with the main theoretical result of the paper ensuring *soundness* and *completeness* for concurrent $CFLP(\mathcal{D})$-derivations with respect to the declarative semantics of the $CFLP(\mathcal{D})$ scheme formalized in [10] by means of a *Constraint Rewriting Logic* $CRWL(\mathcal{D})$. In this result, we use the semantic notions of *answer* and *solution* of a goal G with respect to a domain $\mathcal{D}$ or a program $\mathcal{P}$, formally defined in [5] as logical proofs in $CRWL(\mathcal{D})$. More details on the proof and the technical notations $Sol_{\mathcal{D}}(G)$ and $Sol_{\mathcal{P}}(G)$ used in this theorem can be found in **Appendix** and [5].

Theorem 1. (**Soundness and Completeness**) *Let $S \square \sigma$ be an answer of G.*

(a) **Soundness:** *If $G \vdash^* \|_{i=1}^{k} G_i$ is a concurrent derivation from G of a finite number k of goals G_i, for each $G_i \equiv \square \square S_i \square \sigma_i$ a solved goal, $S_i \square \sigma_i$ is an answer of the initial goal G. Formally, $Sol_{\mathcal{D}}(G_i) \subseteq Sol_{\mathcal{P}}(G)$.*

(b) **Completeness:** *There exists a concurrent derivation $G \vdash^* \|_{i=1}^{k} G_i$, ending with a finite number k of solved goals $G_i \equiv \square \square S_i \square \sigma_i$, that covers all the solutions of the initial answer $S \square \sigma$. Formally, $Sol_{\mathcal{P}}(G) \subseteq \bigcup_{i=1}^{k} Sol_{\mathcal{D}}(G_i)$.*

5 Implementation on the $\mathcal{TOY}$ System

In order to obtain an implementation of our concurrent operational model for constraint functional logic programming, we have adapted the current version of the $\mathcal{TOY}$ system [12] over the constraint finite domain $\mathcal{FD}$, connecting it with the *C* programming language to use native *C* libraries for threading and processes support. We have built a *C* module for evaluating goals by means of the concurrent coordination of constraint solving and demand-driven narrowing, another one for managing the constraint store and the pool of goal solving processes, and a main module to connect all the *C* modules with the *SICStus Prolog* implementation underlying the $\mathcal{TOY}(\mathcal{FD})$ system and the interface with the *SICStus Prolog* $\mathcal{FD}$ constraint solver in demand of these *C* modules.

The design of this prototype tries to modify the least of the current $\mathcal{TOY}(\mathcal{FD})$ system, leaving as many new features as possible in the *C* modules. At the *Prolog* level, we have adapted the definitional trees generation and the predicates to solve goals for the new constrained definitional trees presented in **Section 3**. Then, the access to the constraint store is performed with calls to the connection module that interacts with the *C* module managing processes; goal solving predicates are modified to operate in demand of the *C* module that solves goals.

In this prototype implementation, we wanted to evaluate the practicality and scalability of our approach without designing a whole new system. However, the architecture could be improved in a system designed from the scratch. In particular, the connection interface between *C* and *SICStus Prolog* modules imposes an overhead on the system that could be easily avoided with an improved alternative design. For this reason, we have not considered the time taken by the interface modules when evaluating the performance of our concurrent system.

In **Table 1.** we have evaluated our concurrent prototype with respect to the current implementation of the $\mathcal{TOY}(\mathcal{FD})$ system, the *PAKCS* system and the *SICStus Prolog* implementation. We are comparing the execution times expressed in milliseconds for all the programs of the benchmark. More information about these systems and a precise description of the involved benchmarks solving classical $C(F)LP(\mathcal{FD})$ problems can be found in [4], where the experimental results comparing the non-concurrent version of $\mathcal{TOY}(\mathcal{FD})$ with the other related $C(F)LP(\mathcal{FD})$ systems were originally published. We note that our prototype has a behavior that is comparable to the $\mathcal{TOY}(\mathcal{FD})$ system in every program considered in the benchmark, both in performance and scalability measures, even considering the fact that both architectures are different.

Benchmark [4]	***Concurrent*** $\mathcal{TOY}(\mathcal{FD})$	$\mathcal{TOY}(\mathcal{FD})$	PAKCS	*SICStus*
equation (10)	**15**	20	50	10
equation (20)	**25**	30	55	15
magic (64)	**85**	90	150	80
magic (100)	**205**	220	310	195
magic (200)	**855**	870	1480	850
pythagoras	**40**	50	80	10
queens (8)	**10**	10	15	5
queens (16)	**20**	20	50	8
queens (30)	**130**	150	190	25
suudoku	**10**	10	20	10

Table 1. Concurrent implementation on the $\mathcal{TOY}(\mathcal{FD})$ system [12] vs. other related $C(F)LP(\mathcal{FD})$ systems [4] (execution times expressed in milliseconds).

6 Conclusions and Future Work

The set of transformation rules presented in **Section 4** provides a *sound* and *complete* operational model to describe a concurrent $CFLP(\mathcal{D})$ scheme as a novel generalization of the classical $CLP(\mathcal{D})$ scheme useful for concurrent functional and constraint logic programming. We have implemented all these rules in a prototype version of the $\mathcal{TOY}(\mathcal{FD})$ system, an implementation of the $CFLP(\mathcal{FD})$ instance on finite domain constraints. Our preliminary performance analysis in **Table 1.** has confirmed that our prototype implementation may be competitive with respect to other existing related systems. However, we still need to continue investigating other practical instances of constraint domains (e.g., linear and non-linear arithmetic constraints over real numbers) and the cooperative integration of more efficient constraint solving methods into our concurrent system (e.g., based on the *ILOG CP* technology [1] or using declarative modeling languages such as *OPL*).

References

1. I. Castiñeiras and F. Sáenz-Pérez. *Integrating ILOG CP Technology into* $\mathcal{TOY}$. In Proc. WFLP'09, pages 27-43, 2009.

2. *Curry*. `http://www-ps.informatik.uni-kiel.de/currywiki/`.
3. R. Echahed and W. Serwe. *Defining Actions in Concurrent Declarative Programming*. In Electr. Notes Theor. Comput. Sci. 64, pages 176-194, 2002.
4. A. J. Fernández et. al. *Constraint functional logic programming over finite domains*. Journal of TPLP 7(5), pp. 537–582, 2007.
5. F.J. López-Fraguas, M. Rodríguez-Artalejo, and R. del Vado. *A Lazy Narrowing Calculus for Declarative Constraint Programming*. In PPDP'04, pp. 43–54, 2004.
6. M. Hanus. *Multiparadigm Declarative Languages*. ICLP'07, pp. 45-75, 2007.
7. M. Hanus. *A Unified Computation Model for Functional and Logic Programming*. In Proc. POPL'97, pages 80-93, 1997.
8. J. Jaffar, J.L. Lassez. *Constraint logic programming*. POPL'87, pp. 111-119, 1987.
9. M. Marin, T. Ida, and W. Schreiner. *CFLP: A Mathematica Implementation of a Distributed Constraint Solving System*. The Math. Journal 8(2), pp. 287-300, 2001.
10. F.J. López, M. Rodríguez, and R. del Vado. *A new generic scheme for functional logic programming with constraints*. Journal of HOSC 20 (1-2), pp. 73-122, 2007.
11. V. Saraswat, M. Rinard. *Concurrent constraint programming*. POPL'90, p. 232-245.
12. *$\mathcal{TOY}$: A Constraint Functional Logic System*. Available at `toy.sourceforge.net`.
13. R. del Vado. *A demand-driven narrowing calculus with overlapping definitional Trees*. In PPDP 2003, ACM, pp. 253-263, 2003.
14. `http://gpd.sip.ucm.es/FLOPS2012/examples.pdf`.

Appendix

This appendix presents a transformation algorithm for the pre-runtime construction of constrained definitional trees in **Section 3**, the goal transformation rules for suspensions to complete the concurrent operational semantics given in **Section 4**, and the mathematical proof of the main result of the paper, **Theorem 1**.

Algorithmic Construction of Constrained Definitional Trees

We use a *transformation algorithm* for the pre-runtime construction of constrained definitional trees from the functions defined in a program. Let us start by defining some auxiliary notions.

Definition 1 (Demanded Positions). *Let $\mathcal{P}$ be a program. A* **generic call pattern** *is any call pattern of the form $f\,\overline{X}_n$, where $\overline{X}_n$ are different variables. Let τ a call pattern. We say that:*

(1) A program rule $(l \to r \Leftarrow P \,\square\, C) \in \mathcal{P}$ is **matched** *by τ if there is a substitution σ with $l \equiv \tau\sigma$ (in this case, we write $\tau \preceq l$).*

(2) $p \in VPos\,(\tau)$ (i.e., the set of positions corresponding to variables in τ) is **demanded by the left-hand side** *l of a program rule in $\mathcal{P}$ which is matched by τ if and only if l has a passive symbol at position p.*

(3) $p \in VPos\,(\tau)$ is **demanded** *if and only if p is demanded by the left-hand side of a program rule in $\mathcal{P}$ which is matched by τ.*

(4) $p \in VPos(\tau)$ *is* **uniformly demanded** *if and only if* p *is demanded by every left-hand side of a program rule in* $\mathcal{P}$ *which is matched by* τ.

The *transformation algorithm* is performed by a function named $cDTT$ (i.e., *constrained Definitional Tree Transformation*) that takes a program $\mathcal{P}$ and returns an equivalent program with the corresponding constrained definitional trees. For transforming programs we simply need to apply this algorithm to each defined function $f \in DF^n$ of the program with the initial call

$$cDTT(\mathcal{P}) = \bigcup_{f \in DF} cDTT(f\,\overline{X}_n, \mathcal{P}_f)$$

where $f\,\overline{X}_n$ is a generic call pattern for each $f \in DF^n$ and $\mathcal{P}_f$ is the subset of $\mathcal{P}$ consisting of those rules defining f (obviously, $\mathcal{P} = \bigcup_{f \in DF} \mathcal{P}_f$). The recursive calls will have the form $cDTT(\tau, \mathcal{P})$, where τ is a call pattern and $\mathcal{P}$ is a set of program rules which are matched by τ. We distinguish the following cases:

(1) Some position in $VPos(\tau)$ **is uniformly demanded:**

Let $p \in VPos(\tau)$ be the leftmost such position and let X be the variable at position p in τ. Let $h_1, \ldots, h_m$ be the passive symbols occurring at position p in the left-hand sides of program rules of $\mathcal{P}$ matched by τ. We define $cDTT(\tau, \mathcal{P}) = \bigcup_{1 \leq i \leq m} cDTT(\tau_i, \mathcal{P}_i)$, where:

- $\tau_i = \tau\{X \mapsto h_i \overline{X}_{r_i}\}$ with $\overline{X}_{r_i}$ new variables and r_i is the arity of h_i.
- $\mathcal{P}_i = \{(l \to r \Leftarrow P \,\square\, C) \in \mathcal{P} \mid l \preceq \tau_i\}$.

Then $\mathcal{T}_\tau$ has the form $\underline{case}(\tau, X, \mathit{flex}/\mathit{rigid}, [\mathcal{T}_{\tau_1}, \ldots, \mathcal{T}_{\tau_m}])$, where the trees $\mathcal{T}_{\tau_i}$ are recursively created.

(2) No position in $VPos(\tau)$ **is demanded:**

Then it must be the case that all the left-hand sides of $\mathcal{P}$ which are matched by τ are variants of τ and we can define $cDTT(\tau, \mathcal{P}) = \mathcal{P}$. Moreover, $\mathcal{T}_\tau$ has the structure $\underline{rule}(\tau \to r_1 \Leftarrow P_1 \,\square\, C_1 \| \ldots \| r_m \Leftarrow P_m \,\square\, C_m)$.

(3) Some position in $VPos(\tau)$ **is demanded, but no one is uniformly demanded:**

Let $p \in VPos(\tau)$ be the leftmost such position. We take $\mathcal{P}_p^+ = \{R \in \mathcal{P} \mid$ the left-hand side of R demands $p\}$ and $\mathcal{P}_p^- = \mathcal{P} \backslash \mathcal{P}_p^+$. We consider two new function symbols, f_1, f_2 with the same arities than f (the defined function symbol in the root of τ), and we define $cDTT(\tau, \mathcal{P}) = cDTT(\tau_1, \mathcal{P}_1) \cup cDTT(\tau_2, \mathcal{P}_2) \cup \{\tau \to \tau_1 \,\|\, \tau_2\}$, where:

- τ_i is the new call pattern built changing the occurrence of f in the root of τ by the new function symbol f_i.
- $\mathcal{P}_1$ (respectively $\mathcal{P}_2$) is the set of program rules built from $\mathcal{P}_p^+$ (respectively $\mathcal{P}_p^-$) by changing the occurrence of f in the left-hand sides of their program rules by the new function symbol f_1 (respectively f_2).

The definitional tree $\mathcal{T}_\tau$ has the form $\underline{rule}(\tau \to \tau_1 \,\|\, \tau_2)$, where $\mathcal{T}_{\tau_1}$ and $\mathcal{T}_{\tau_2}$ are recursively created.

Example 1. We illustrate with an example the application of the transformation algorithm. Consider the program $\mathcal{P}$ to compute *Fibonacci numbers* given in **Section 2**:

$$\begin{array}{l} \mathit{fib}\,(0) \;\to 1 \\ \mathit{fib}\,(1) \;\to 1 \\ \mathit{fib}\,(N) \to \mathit{fib}\,(N-1) + \mathit{fib}\,(N-2) \Leftarrow N \geq 2 \end{array}$$

We apply the transformation algorithm:

$$cDTT\,(\mathcal{P}) = cDTT\,(\mathit{fib}\,(N), \mathcal{P}) =$$

N is in a demanded position, but no uniformly. We consider:

$$\begin{array}{l} \mathcal{P}^+ = \{\mathit{fib}\,(0) \to 1,\, \mathit{fib}\,(1) \to 1\} \\ \mathcal{P}^- = \{\mathit{fib}\,(N) \to \mathit{fib}\,(N-1) + \mathit{fib}\,(N-2) \Leftarrow N \geq 2\} \end{array}$$

$$\begin{array}{l} \mathcal{P}_1 = \{\mathit{fib}_1\,(0) \to 1,\, \mathit{fib}_1\,(1) \to 1\} \\ \mathcal{P}_2 = \{\mathit{fib}_2\,(N) \to \mathit{fib}\,(N-1) + \mathit{fib}\,(N-2) \Leftarrow N \geq 2\} \end{array}$$

$$\mathcal{T}_{\mathit{fib}} \equiv \underline{rule}\,(\mathit{fib}\,(N) \;\to\; \mathit{fib}_1(N) \;||\; \mathit{fib}_2(N))$$

$$= cDTT\,(\mathit{fib}_1\,(N), \mathcal{P}_1) \cup cDTT\,(\mathit{fib}_2\,(N), \mathcal{P}_2) \cup \{\mathit{fib}\,(N) \;\to\; \mathit{fib}_1(N) \,\|\, \mathit{fib}_2(N)\} =$$

- $cDTT\,(\mathit{fib}_1\,(N), \mathcal{P}_1) = cDTT\,(\mathit{fib}_1\,(0), \mathcal{P}_{11}) \cup cDTT\,(\mathit{fib}_1\,(1), \mathcal{P}_{12})$ with $\mathcal{P}_{11} = \{\mathit{fib}_1\,(0) \to 1\}$ and $\mathcal{P}_{12} = \{\mathit{fib}_1\,(1) \to 1\}$, because N is in a uniformly demanded position. Finally, $cDTT\,(\mathit{fib}_1\,(0), \mathcal{P}_{11}) = \mathcal{P}_{11}$ and $cDTT\,(\mathit{fib}_1\,(1), \mathcal{P}_{12}) = \mathcal{P}_{12}$, because no position is demanded. Therefore

$$\mathcal{T}_{\mathit{fib}_1} \equiv \underline{case}\,(\mathit{fib}_1(N), N, \mathit{flex}/\mathit{rigid}, [\,\underline{rule}\,(\mathit{fib}_1\,(0) \to 1), \underline{rule}\,(\mathit{fib}_1\,(1) \to 1)])$$

- $cDTT\,(\mathit{fib}_2\,(N), \mathcal{P}_2) = \mathcal{P}_2$, because no position is demanded. Therefore

$$\mathcal{T}_{\mathit{fib}_2} \equiv \underline{rule}\,(\mathit{fib}_2(N) \;\to\; \mathit{fib}\,(N-1) + \mathit{fib}\,(N-2) \;\Leftarrow\; N \geq 2)$$

$$\begin{array}{ll} = & \{\mathit{fib}_1\,(0) \to 1\} \cup \{\mathit{fib}_1\,(1) \to 1\} \cup \\ & \{\mathit{fib}_2(N) \;\to\; \mathit{fib}\,(N-1) + \mathit{fib}\,(N-2) \;\Leftarrow\; N \geq 2\} \cup \\ & \{\mathit{fib}\,(N) \;\to\; \mathit{fib}_1(N) \,\|\, \mathit{fib}_2(N)\} \end{array}$$

Proofs Omitted from the Paper

The final subsection of the **Appendix** presents the proof of the main result of the paper, **Theorem 1**, stating the *soundness* and *completeness* of concurrent $CFLP(\mathcal{D})$-derivations with respect to $CRWL(\mathcal{D})$-semantics [10] formalized

EL ELimination

$$e \to X, P \square C \square S \square \sigma \Vdash_{\mathbf{EL}} P \square C \square S \square \sigma \qquad \text{if } X \notin Var(P \square C \square S \square \sigma).$$

SS Simple Suspension

$$t \to X, P \square C \square S \square \sigma \Vdash_{\mathbf{SS}} (P \square C \square S)\sigma_0 \square \sigma\sigma_0 \qquad \text{if } t \in Pat(\mathcal{U}) \text{ and } \sigma_0 = \{X \mapsto t\}.$$

IM IMitation

$$h\overline{e}_m \to X, P \square C \square S \square \sigma \Vdash_{\mathbf{IM}} \left\{ \begin{array}{ll} \|_{i=1}^{m} (e_i \to X_i, P \square C \square S)\sigma_0 \square \sigma\sigma_0 & \\ & \text{if } X \in DVar_{\mathcal{D}}(P \square S) \\ \circlearrowleft & \text{if } X \notin DVar_{\mathcal{D}}(P \square S) \end{array} \right\}$$

if $h\overline{e}_m \notin Pat(\mathcal{U})$ is passive, and $\sigma_0 = \{X \mapsto h\overline{X}_m\}$ with $\overline{X}_m$ new variables.

PF Primitive Function

$$p\overline{e}_n \to X, P \square C \square S \square \sigma \Vdash_{\mathbf{PF}} \left\{ \begin{array}{ll} \|_{i=1}^{n} e_i \to X_i, P \square C \square p\overline{X}_n = X, S \square \sigma & \\ & \text{if } X \in DVar_{\mathcal{D}}(P \square S) \\ \circlearrowleft & \text{if } X \notin DVar_{\mathcal{D}}(P \square S) \end{array} \right\}$$

if $p \in PF^n$, and $\overline{X}_n$ are new variables.

FV Functional Variable

$$F\overline{e}_q \to X, P \square C \square S \square \sigma \Vdash_{\mathbf{FV}} \left\{ \begin{array}{ll} \|_h (h\overline{X}_p\overline{e}_q \to X, P \square C \square S)\sigma_0 \square \sigma\sigma_0 & \\ & \text{if } F \notin PVar(P) \text{ and } X \in DVar_{\mathcal{D}}(P \square S) \\ \circlearrowleft & \text{otherwise} \end{array} \right\}$$

if $q > 0$, $\sigma_0 = \{F \mapsto h\overline{X}_p\}$, and $\overline{X}_p$ are new variables.

Fig. 4. Rules for concurrency in demand-driven narrowing: Suspensions.

by means of *logical proofs* $\vdash_{\mathcal{D}}$. Within this mathematical proof we assume, for each concurrent goal transformation rule given in **Section 4** and the *suspension* rules given in **Fig. 4** (these rules can be easily adapted from other works in $CFLP(\mathcal{D})$[4] and for this reason have been omitted from the paper), that goals G are exactly as they appear in the paper with an explicit prefix $\exists U.\, G$ of existentially quantified variables $U \equiv evar\,(G)$ (see [5] for more details). We start by defining some auxiliary notions.

Definition 2 (Answers and Solutions). *An* **answer** *for a goal* $G \equiv P \square C \square S \square \sigma$ *and a given program* $\mathcal{P}$*, must have the form* $\Pi \square \theta$*, where* Π *is a finite conjunction of primitive constraints,* θ *is an idempotent substitution such that* $Dom\,(\theta) \cap Var\,(\Pi) = \emptyset$*, and there is some substitution* $\theta' =_{\setminus evar(G)} \theta$ *(i.e.,* θ' *is equal to* θ *except for the set of variables* $evar(G)$*) fulfilling the following three conditions:*

(1) $\mathcal{P} \vdash_{\mathcal{D}} (P \square C)\theta' \Leftarrow \Pi$ *in* $CRWL(\mathcal{D})$ *[5] (in this paper, for demanded productions* $(< \tau, \mathcal{T} > \to R) \in P$*, we consider just* $\mathcal{P} \vdash_{\mathcal{D}} \tau\theta' \to \theta'(R) \Leftarrow \Pi$*, because constrained definitional trees are only used to control the computation),*

(2) $\Pi \models_{\mathcal{D}} S\theta'$ *(i.e.,* $Sol_{\mathcal{D}}(\Pi) \subseteq Sol_{\mathcal{D}}(S\theta')$*, where* $Sol_{\mathcal{D}}(\Pi)$*, analogously for* $Sol_{\mathcal{D}}(S\theta')$*, represents the set of substitutions* μ *that satisfy* $\Pi\mu$ *on the constraint domain* $\mathcal{D}$*),*

(3) $X\theta' \equiv t\theta'$ *for each binding* $\{X \mapsto t\} \in \sigma$ *(abbreviated as* $\theta' \in Sol(\sigma)$*).*

We write $Ans_{\mathcal{P}}(G)$ *for the set of all answers for* G*. We say that a substitution* θ *is a* **solution** *of* G *if and only if* $\emptyset \square \theta \in Ans_{\mathcal{P}}(G)$*. We write* $Sol_{\mathcal{P}}(G)$ *for the set of all solutions for* G*. For the particular case of solved goals* $G \equiv \square \square S \square \sigma$ *we use the notation* $Sol_{\mathcal{D}}(G)$*.*

The first result proves correctness of a single transformation step. It says that transformation steps preserve admissibility of goals, fail only in case of unsatisfiable goals and do not introduce new solutions.

Lemma 1 (Correctness Lemma). *The concurrent goal transformation steps do not introduce new solutions: If* $G \Vdash \|_{i=1}^{k} G_i$ *and* $\Pi \square \theta \in Ans_{\mathcal{P}}(G_i)$ *then* $\Pi \square \theta \in Ans_{\mathcal{P}}(G)$*. Thus,* $Sol_{\mathcal{P}}(G_i) \subseteq Sol_{\mathcal{P}}(G)$ *for all* $1 \leq i \leq k$*.*

Proof. We proceed by considering concurrent goal transformation rules $G \Vdash_{\mathbf{R}} \|_{i=1}^{k} G_i$ one by one (we use G' instead of G_i if $k = 1$):

SS Assume that $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\setminus evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\sigma_0\hat{\theta}$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$ and $\mathcal{P} \vdash_{\mathcal{D}} (P \ \square \ C)\sigma_0\hat{\theta} \Leftarrow \Pi$. We define $\hat{\theta}'(X) = t\hat{\theta}$ y $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for $Y \neq X$. It holds that $\sigma_0\hat{\theta} = \hat{\theta}'$. Then, $\Pi \models_{\mathcal{D}} S\hat{\theta}'$ and $\mathcal{P} \vdash_{\mathcal{D}} (P \ \square \ C)\hat{\theta}' \Leftarrow \Pi$. Since $X \in pvar(P)$, we have $X \notin var(\sigma)$ by the admissibility condition **SL** [5], an $Y\hat{\theta}' \equiv Y\hat{\theta} \equiv s\hat{\theta} \equiv s\hat{\theta}'$ for each $Y \mapsto s \in \sigma$. Moreover, since $X \notin var(t)$, $X\hat{\theta}' \equiv t\theta \equiv t\hat{\theta}'$. Then, $\Pi \models_{\mathcal{D}} t\hat{\theta}' \sqsupseteq X\hat{\theta}'$, and using the *Approximation Property*, we also have $\mathcal{P} \vdash_{\mathcal{D}} (t \to X)\hat{\theta}' \Leftarrow \Pi$. Finally, since $\hat{\theta} =_{\setminus \{X\}} \hat{\theta}'$ we conclude $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G)$.

IM Assume that $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\setminus evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\sigma_0\hat{\theta}$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \ \square \ C)\sigma_0\hat{\theta} \Leftarrow \Pi$ and $\mathcal{T}_i \equiv \mathcal{P} \vdash_{\mathcal{D}} (e_i \to X_i)\sigma_0\hat{\theta} \Leftarrow \Pi$ for each $1 \leq i \leq m$. Since $X \notin var(e_i)$ by the admissibility condition **NC** [5] and $\overline{X}_m$ are new variables in G', we also have $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} (e_i \to X_i)\hat{\theta} \Leftarrow \Pi$. Now, we define $\hat{\theta}'(X) = h\overline{X_m\hat{\theta}}$, $\hat{\theta}'(X_i) = X_i$ for all $1 \leq i \leq m$ and $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for all $Y \notin \{X, \overline{X}_m\}$. It holds that $\sigma_0\hat{\theta} =_{\setminus \{\overline{X}_m\}} \hat{\theta}'$. Since $\overline{X}_m$ are new variables in G', $\Pi \models_{\mathcal{D}} S\hat{\theta}'$, $\mathcal{P} \vdash_{\mathcal{D}} (P \ \square \ C)\hat{\theta}' \Leftarrow \Pi$ and $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta}' \to X_i\hat{\theta} \Leftarrow \Pi$. Furthermore, since X is a produced variable in G, $X \notin var(\sigma)$ by the admissibility condition **SL** [5], and then $Y\hat{\theta}' \equiv Y\hat{\theta} \equiv s\hat{\theta} \equiv s\hat{\theta}'$ for each $Y \mapsto s \in \sigma$. Finally, $\mathcal{T} \equiv \mathbf{DC}\ (h\overline{e_m\hat{\theta}'} \to h\overline{X_m\hat{\theta}}) \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_m])$ is a proof tree for $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (h\overline{e}_m \to X)\hat{\theta}' \Leftarrow \Pi$. Therefore, since $\hat{\theta} =_{\setminus \{\overline{X}_m\}} \hat{\theta}'$ we conclude $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G)$.

EL Assume that $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\setminus evar(G')} \theta$ such that

$\Pi \models_{\mathcal{D}} S\hat{\theta}, Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \square C)\hat{\theta} \Leftarrow \Pi$. We define $\hat{\theta}'(X) = \bot$ and $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for all $Y \neq X$ (note that $\hat{\theta}'$ is well defined, because X is a produced but not demanded variable in G). It holds that $\hat{\theta} =_{\setminus\{X\}} \hat{\theta}'$. Since $X \notin var(P \square C \square S \square \sigma)$, we directly have $\Pi \models_{\mathcal{D}} S\hat{\theta}', Y\hat{\theta}' \equiv Y\hat{\theta} \equiv s\hat{\theta} \equiv s\hat{\theta}'$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \square C)\hat{\theta}' \Leftarrow \Pi$. Furthermore, $\mathcal{T} \equiv \mathbf{TI}\ (e\hat{\theta}' \to \bot \Leftarrow \Pi, [\,])$ is a proof tree for $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (e \to X)\hat{\theta}' \Leftarrow \Pi$. Therefore, since $\hat{\theta} =_{\setminus\{X\}} \hat{\theta}'$ we conclude $\Pi \square \theta \in Ans_{\mathcal{P}}(G)$.

PF Assume that $\Pi \square \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\setminus evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $\Pi \models_{\mathcal{D}} (p\bar{t}_n \to! X)\hat{\theta}$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \square C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e_j \to X_j)\hat{\theta} \Leftarrow \Pi$ for each $1 \leq j \leq q$ (if $q = 0$ these proofs are omitted). We define $\hat{\theta}'(X_j) = X_j$ for each $1 \leq j \leq q$ and $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for each $Y \notin \{X_1, \ldots, X_q\}$. It holds that $\hat{\theta}' =_{\setminus\{\overline{X}_q\}} \hat{\theta}$. Since $\overline{X}_q$ are new variables, $\Pi \models_{\mathcal{D}} S\hat{\theta}'$, $Y\hat{\theta}' \equiv Y\hat{\theta} \equiv s\hat{\theta} \equiv s\hat{\theta}'$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \square C)\hat{\theta}' \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} e_j\hat{\theta}' \to X_j\hat{\theta} \Leftarrow \Pi$ for each $1 \leq j \leq q$. Since $e_j \notin Pat_{\bot}(\mathcal{U})$ for all $1 \leq j \leq q$, we have $\mathcal{P} \vdash_{\mathcal{D}} e_j\hat{\theta}' \to t_j\hat{\theta} \Leftarrow \Pi$ for each $1 \leq j \leq q$. Furthermore, for each $e_i \in Pat_{\bot}(\mathcal{U})$ we know that $e_i \equiv t_i$ and trivially $\Pi \models_{\mathcal{D}} e_i\hat{\theta}' \sqsupseteq t_i\hat{\theta}$. By the *Approximation Property* [10] we can obtain proof trees $\mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta}' \to t_i\hat{\theta} \Leftarrow \Pi$. Therefore, we have proof trees $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta}' \to t_i\hat{\theta} \Leftarrow \Pi$ for each $1 \leq i \leq n$. Then, we can build a proof tree $\mathcal{T} \equiv \mathbf{PF}\ (p\overline{e}_n\hat{\theta}' \to X\hat{\theta} \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n])$ ($X\hat{\theta} \neq \bot$ because $X \in dvar_{\mathcal{D}}(G')$ and $\Pi \square \theta \in Ans_{\mathcal{P}}(G')$ with $\hat{\theta} =_{\setminus evar(G)} \theta$, and by the *Demand Lemma* [10] $\hat{\theta}(X) \neq \bot$). Finally, since $\overline{X}_q$ are new variables distinct to X, $\hat{\theta}(X) \equiv \hat{\theta}'(X)$ and we have that $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (p\overline{e}_n \to X)\hat{\theta}' \Leftarrow \Pi$. Therefore, since $\hat{\theta} =_{\setminus\{\overline{X}_q\}} \hat{\theta}'$ we conclude $\Pi \square \theta \in Ans_{\mathcal{P}}(G)$.

DT$_1$ Assume that $\Pi \square \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\setminus evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \square C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (f\ \overline{e}_n \to X)\hat{\theta} \Leftarrow \Pi$ (the definitional tree $\mathcal{T}_{f\overline{X}_n}$ is only used to control the computation). Then, directly we conclude $\Pi \square \theta \in Ans_{\mathcal{P}}(G)$.

DT$_2$ Assume that $\Pi \square \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\setminus evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \square C)\hat{\theta} \Leftarrow \Pi$, $\mathcal{P} \vdash_{\mathcal{D}} (X'\ \overline{a}_k \to X)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (f\ \overline{e}_n \to X')\hat{\theta} \Leftarrow \Pi$ (the definitional tree $\mathcal{T}_{f\overline{X}_n}$ is used only to control the computation). By the *Demand Lemma* [10] we have that $\hat{\theta}(X') \neq \bot$, because X' is a demanded variable (due to the conditions of the rule $X \in dvar_{\mathcal{D}}(G)$, and $X \in dvar_{\mathcal{D}}(G')$ since P and S in G are not changed, by definition of demanded variable X' is also demanded in G', according to the admissibility condition **DT** for G'). Therefore, we have that $\mathcal{T}' \equiv \mathbf{DF}_{\mathcal{P}}\ ((f\overline{e}_n \to X')\hat{\theta} \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n, \mathcal{T}_c, \mathcal{T}_r])$ is a proof tree for $\mathcal{T}' : \mathcal{P} \vdash_{\mathcal{D}} (f\ \overline{e}_n \to X')\hat{\theta} \Leftarrow \Pi$, using $(f\ \bar{t}_n \to r \Leftarrow P' \square C') \in [\mathcal{P}]_{\bot}$ and where $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta} \to t_i \Leftarrow \Pi$ $(1 \leq i \leq n)$, $\mathcal{T}_c : \mathcal{P} \vdash_{\mathcal{D}} P' \square C' \Leftarrow \Pi$ and $\mathcal{T}_r : \mathcal{P} \vdash_{\mathcal{D}} r \to \hat{\theta}(X') \Leftarrow \Pi$. Now, since we also have $\mathcal{T}_s : \mathcal{P} \vdash_{\mathcal{D}} \hat{\theta}(X')\ \overline{a_k\hat{\theta}} \to \hat{\theta}(X) \Leftarrow \Pi$ and $\hat{\theta}(X') \in Pat_{\bot}(\mathcal{U})$, we can build the proof

tree $\mathcal{T} \equiv \mathbf{DF}_{\mathcal{P}}\ ((f\ \overline{e}_n\ \overline{a}_k \to X)\hat{\theta} \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n, \mathcal{T}_c, \mathcal{T}_r, \mathcal{T}_s\ ])\ (k > 0)$ for $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (f\ \overline{e}_n\ \overline{a}_k \to X)\hat{\theta} \Leftarrow \Pi\ (k > 0)$. Therefore, we conclude $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G)$.

FV Assume that $\Pi\ \square\ \hat{\theta} \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\backslash evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\sigma_0\hat{\theta}$, $F\hat{\theta} \equiv h\overline{X_m\hat{\theta}}$ for $F \mapsto h\overline{X}_m \in \sigma_0$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P\ \square\ C)\sigma_0\hat{\theta} \Leftarrow \Pi$ and $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (h\ \overline{X}_m\ \overline{a}_k \to X)\sigma_0\hat{\theta} \Leftarrow \Pi$. We define $\hat{\theta}'(X_i) = X_i$ for each $1 \leq i \leq m$ and $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for all $Y \notin \{\overline{X}_m\}$. It holds that $\sigma_0\hat{\theta} =_{\backslash \{\overline{X}_m\}} \hat{\theta}'$. Since $\overline{X}_m$ are new variables, $\Pi \models_{\mathcal{D}} S\hat{\theta}'$, $Y\hat{\theta}' \equiv Y\hat{\theta} \equiv s\hat{\theta} \equiv s\hat{\theta}'$ for each $Y \mapsto s \in \sigma$ and $\mathcal{P} \vdash_{\mathcal{D}} (P\ \square\ C)\hat{\theta}' \Leftarrow \Pi$. Furthermore, since $F \notin \{\overline{X}_m\}$ and $\hat{\theta}'(F) = h\overline{X_m\hat{\theta}}$, we have that $\mathcal{P} \vdash_{\mathcal{D}} (F\ \overline{a}_k \to X)\hat{\theta}' \Leftarrow \Pi$. Therefore, since $\hat{\theta} =_{\backslash \{F, \overline{X}_m\}} \hat{\theta}'$, we conclude $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G)$.

CSS Assume that $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\backslash evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$ and $\mathcal{P} \vdash_{\mathcal{D}} (P\ \square\ C)\hat{\theta} \Leftarrow \Pi$, $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ (the definitional tree is used only to control the computation). Then, we have directly $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G)$.

DP, DR, DI Assume that $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\backslash evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\sigma_0\hat{\theta}$, $Y\hat{\theta} \equiv (h_i\overline{Y}_{m_i})\hat{\theta}$ for $Y \mapsto h_i\overline{Y}_{m_i} \in \sigma_0$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P\ \square\ C)\sigma_0\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\sigma_0\hat{\theta} \Leftarrow \Pi$ (the definitional tree is only used to control the computation). Due to the admissibility condition **NC**, $Y \neq R$. Since $dom(\sigma_0) = \{Y\}$, $R\sigma_0\hat{\theta} = R\hat{\theta}$. Furthermore, since $e|_{pos(X,\tau)} = Y$, $(e|_{pos(X,\tau)})\sigma_0\hat{\theta} = (h_i\overline{Y}_{m_i})\hat{\theta} = Y\hat{\theta}$. Moreover, for all $Z \notin \{R, Y\}$ it holds that $Z\sigma_0\hat{\theta} = Z\hat{\theta}$. Therefore, $\sigma_0\hat{\theta} = \hat{\theta}$. Then, $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P\ \square\ C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$. Therefore, since $evar(G) \subseteq evar(G')$ and $\overline{Y}_{m_i}$ are new variables that not occur in G, $\theta =_{\backslash evar(G)} \hat{\theta}$ and finally we have that $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G)$.

DN Assume that $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\backslash evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P\ \square\ C)\hat{\theta} \Leftarrow \Pi$, $\mathcal{P} \vdash_{\mathcal{D}} (e|_{pos(X,\tau)} \to R')\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e[R']_{pos(X,\tau)} \to R\ )\hat{\theta} \Leftarrow \Pi$. Since $\hat{\theta} \in Sub_{\perp}(\mathcal{U})$, we also have $\mathcal{P} \vdash_{\mathcal{D}} e\hat{\theta}|_{pos(X,\tau)} \to \hat{\theta}(R') \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} e\hat{\theta}[\hat{\theta}(R')]_{pos(X,\tau)} \to \hat{\theta}(R) \Leftarrow \Pi$ are provable, where $\hat{\theta}(R') \in Pat_{\perp}(\mathcal{U})$. By the *Splitting Property*, $\mathcal{P} \vdash_{\mathcal{D}} e\hat{\theta} \to \hat{\theta}(R) \Leftarrow \Pi$, and then, $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$. Therefore, since $evar(G) \subseteq evar(G')$ and R' is a new variable, $\hat{\theta} =_{\backslash evar(G)} \theta$ and finally we have that $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G)$.

RRA Assume that $\Pi\ \square\ \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\backslash evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Y\hat{\theta} \equiv s\hat{\theta}$ for each $Y \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P\ \square\ C)\hat{\theta} \Leftarrow \Pi$, $\mathcal{T}_{c_i} : \mathcal{P} \vdash_{\mathcal{D}} (P_i\ \square\ C_i)\sigma_c\hat{\theta} \Leftarrow \Pi$, $\mathcal{T}_{r_i} : \mathcal{P} \vdash_{\mathcal{D}} (r_i\sigma_c \to R)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (\sigma_f(R_j) \to R_j)\hat{\theta} \Leftarrow \Pi$ for each $1 \leq j \leq m$. By the *Demand Lemma*, we have that $\hat{\theta}(R) \neq \perp$ (R is a demanded variable in G and G'). We assume that τ is of the form $f\overline{t}_n$ and

then $e = \tau\sigma = f\overline{t_n\sigma}$ and $e\hat{\theta} = f\overline{t_n\sigma\hat{\theta}}$. Therefore, a proof of $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ uses the $CRWL(\mathcal{D})$-rule $\mathbf{DF}_{\mathcal{P}}$ with a partial instantiation by $\sigma_c\hat{\theta}$ of the program rule $(f\bar{t}_n \to r_i \Leftarrow P_i \;\square\; C_i) \in \mathcal{P}$ (for $1 \leq i \leq k$) such that $\mathcal{T} \equiv \mathbf{DF}_{\mathcal{P}}$ $(f\overline{t_n\sigma\hat{\theta}} \to \hat{\theta}(R) \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n, \mathcal{T}_{c_i}, \mathcal{T}_{r_i}])$ where $\mathcal{T}_j : \mathcal{P} \vdash_{\mathcal{D}} (t_j\sigma \to t_j\sigma_c)\hat{\theta} \Leftarrow \Pi$ for all $1 \leq j \leq n$. On the other hand, for each variable $X \in var(t_j)$, $\mathcal{P} \vdash_{\mathcal{D}} \hat{\theta}(\sigma(X)) \to \hat{\theta}(\sigma_c(X)) \Leftarrow \Pi$ is provable:

- If $X \notin dom_f(\sigma)$, we have $\mathcal{P} \vdash_{\mathcal{D}} \hat{\theta}(\sigma(X)) \to \hat{\theta}(\sigma_c(X)) \Leftarrow \Pi$ because $\sigma(X) = \sigma_c(X)$ and $\Pi \models_{\mathcal{D}} \hat{\theta}(\sigma_c(X)) \sqsupseteq \hat{\theta}(\sigma_c(X))$ with $\hat{\theta}(\sigma_c(X)) \in Pat_{\perp}(\mathcal{U})$.
- If $X \in dom_f(\sigma) = \{R_1, \ldots, R_m\}$, $\hat{\theta}(\sigma(X)) = \hat{\theta}(\sigma_f(X))$ and $\mathcal{P} \vdash_{\mathcal{D}} \hat{\theta}(\sigma_f(X)) \to \hat{\theta}(X)$ $(= \hat{\theta}(\sigma_c(X)))$ $\Leftarrow \Pi$ is provable in $CRWL(\mathcal{D})$ by initial hypothesis.

Since $evar(G) \subseteq evar(G')$ and $\overline{X}$ are new variables, $\hat{\theta} =_{\setminus evar(G)} \theta$ and then $\Pi \;\square\; \theta \in Ans_{\mathcal{P}}(G)$.

CS Assume that $\Pi_i \;\square\; \theta_i \in Ans_{\mathcal{P}}(G_i)$ $(1 \leq i \leq k)$. There exists $\hat{\theta}_i =_{\setminus evar(G_i)} \theta_i$ such that $\Pi_i \models_{\mathcal{D}} S_i\hat{\theta}_i$, $X\hat{\theta}_i \equiv t\hat{\theta}_i$ for each $X \mapsto t \in \sigma_i$, $Y\hat{\theta}_i \equiv s\hat{\theta}_i$ for each $Y \mapsto s \in \sigma$ and $\mathcal{P} \vdash_{\mathcal{D}} (P \,\square\, C)\sigma_i\hat{\theta}_i \Leftarrow \Pi_i$. We define $\hat{\theta'}_i(Y) = Y$ for all $Y \in \overline{Y}_i \setminus dom(\sigma_i)$ and $\hat{\theta'}_i(Z) = \hat{\theta}_i(Z)$ for $Z \in (\setminus\{\overline{Y}_i\}) \cup dom(\sigma_i)$. Since $X\hat{\theta}_i \equiv t\hat{\theta}_i$ for each $X \mapsto t \in \sigma_i$, it holds that $\sigma_i\hat{\theta}_i =_{(\setminus\{\overline{Y}_i\}) \cup dom(\sigma_i)} \hat{\theta'}_i$. Since $\overline{Y}_i$ are new variables, $Y\hat{\theta'}_i \equiv Y\hat{\theta}_i \equiv s\hat{\theta}_i \equiv s\hat{\theta'}_i$ for each $Y \to s \in \sigma$ and $\mathcal{P} \vdash_{\mathcal{D}} (P \,\square\, C)\hat{\theta'}_i \Leftarrow \Pi_i$. Now, we prove $\Pi_i \models_{\mathcal{D}} S\hat{\theta'}_i$. Let $\mu \in Sol_{\mathcal{D}}(\Pi_i)$. Since $\Pi_i \models_{\mathcal{D}} S_i\hat{\theta}_i$, we have $\mu \in Sol_{\mathcal{D}}(S_i\hat{\theta}_i)$. Hence, $\hat{\theta}_i\mu \in Sol_{\mathcal{D}}(S_i)$. Furthermore, since $X\hat{\theta}_i \equiv t\hat{\theta}_i$ for each $X \mapsto t \in \sigma_i$ we also have that $X\hat{\theta}_i\mu \equiv t\hat{\theta}_i\mu$ for each $X \mapsto t \in \sigma_i$. Therefore, there exists $\hat{\theta'}_i\mu =_{\setminus S} \hat{\theta}_i\mu$ such that $\hat{\theta}_i\mu \in Sol_{\mathcal{D}}(S_i)$ and $X\hat{\theta}_i\mu \equiv t\hat{\theta}_i\mu$ for each $X \mapsto t \in \sigma_i$. By definition, $\hat{\theta}_i\mu \in Sol_{\mathcal{D}}(\,\exists_{\setminus S}.\; S_i \;\square\; \sigma_i)$ $(1 \leq i \leq k)$ and then $\hat{\theta}_i\mu \in \bigcup_{i=1}^{k} Sol_{\mathcal{D}}\;(\exists_{\setminus S}.\; S_i \;\square\; \sigma_i)$. Using the requirement of a constraint solver, it follows that also $\hat{\theta}_i\mu \in Sol_{\mathcal{D}}(S)$. Then, $\mu \in Sol_{\mathcal{D}}(S\hat{\theta}_i)$. Therefore, we conclude $\Pi_i \;\square\; \theta_i \in Ans_{\mathcal{P}}(G)$ for each $1 \leq i \leq k$.

AC Assume that $\Pi \;\square\; \theta \in Ans_{\mathcal{P}}(G')$. There exists $\hat{\theta} =_{\setminus evar(G')} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $\Pi \models_{\mathcal{D}} (p\bar{t}_n \to!\; t)\hat{\theta}$, $X\hat{\theta} \equiv s\hat{\theta}$ for each $X \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \;\square\; C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e_j \to X_j)\hat{\theta} \Leftarrow \Pi$ for each $1 \leq j \leq q$ (if $q = 0$ these proofs are omitted). We define $\hat{\theta'}(X_j) = X_j$ for each $1 \leq j \leq q$ and $\hat{\theta'}(Y) = \hat{\theta}(Y)$ for each $Y \notin \{X_1, \ldots, X_q\}$. It holds that $\hat{\theta'} =_{\setminus\{\overline{X}_q\}} \hat{\theta}$. Since $\overline{X}_q$ are new variables, $\Pi \models_{\mathcal{D}} S\hat{\theta'}$, $X\hat{\theta'} \equiv X\hat{\theta} \equiv s\hat{\theta} \equiv s\hat{\theta'}$ for each $X \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \;\square\; C)\hat{\theta'} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} e_j\hat{\theta'} \to X_j\hat{\theta} \Leftarrow \Pi$ for each $1 \leq j \leq q$. Since $e_j \notin Pat_{\perp}(\mathcal{U})$ for all $1 \leq j \leq q$, we have $\mathcal{P} \vdash_{\mathcal{D}} e_j\hat{\theta'} \to t_j\hat{\theta} \Leftarrow \Pi$ for each $1 \leq j \leq q$. Furthermore, for each $e_i \in Pat_{\perp}(\mathcal{U})$ we know that $e_i \equiv t_i$ and trivially $\Pi \models_{\mathcal{D}} e_i\hat{\theta'} \sqsupseteq t_i\hat{\theta}$. By the *Approximation Property* we obtain proof trees $\mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta'} \to t_i\hat{\theta} \Leftarrow \Pi$. Therefore, we have proof trees $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta'} \to t_i\hat{\theta} \Leftarrow \Pi$ for each $1 \leq i \leq n$. Since $\Pi \models_{\mathcal{D}} (p\bar{t}_n \to!\; t)\hat{\theta}$, we can build a proof tree $\mathcal{T} \equiv \mathbf{AC}\;(p\bar{e}_n\hat{\theta'} \to t\hat{\theta} \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n])$ $(t\hat{\theta} \neq \perp$ because $t \notin \mathcal{V}$ or $t \in dvar_{\mathcal{D}}(G')$ and we can apply the *Demand*

Lemma). Finally, since $\overline{X}_q$ are new variables, $t\hat{\theta} \equiv t\hat{\theta}'$ and we have that $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (p\overline{e}_n \to! \ t)\hat{\theta}' \Leftarrow \Pi$. Therefore, since $\hat{\theta} =_{\setminus\{\overline{X}_q\}} \hat{\theta}'$ we conclude $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G)$.

Finally, we proceed by considering failure rules one by one:

CC Assume that $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G)$. By definition, there exists a substitution $\hat{\theta}$ such that $\hat{\theta} =_{\setminus evar(G)} \theta$ and $< e, \underline{case} \ (\tau, \ X, \ op, \ [\mathcal{T}_1, \ \ldots, \ \mathcal{T}_k]) > \to R$ is $CRWL(\mathcal{D})$-deducible for $e\hat{\theta} \to R\hat{\theta} \Leftarrow \Pi$, using in the last step the $CRWL(\mathcal{D})$-rule $\mathbf{DF}_{\mathcal{P}}$ with a partial instantiation $(f\overline{t}_n \to r \Leftarrow P \ \square \ C) \in [\mathcal{P}]_{\perp}$ ($e\hat{\theta} = f\overline{e_n\hat{\theta}}$ and R is a demanded variable with $\hat{\theta}(R) \neq \perp$ according to the *Demand Lemma*). It follows that $\mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta} \to t_i \Leftarrow \Pi$ is $CRWL(\mathcal{D})$-deducible for all $1 \leq i \leq n$. Since $e|_{pos(X,\tau)} = h \ldots$, $e\hat{\theta}$ has the symbol h in an argument $e_i\hat{\theta}$. Since $\tau \preccurlyeq f\overline{t}_n$, $f\overline{t}_n$ has the symbol h_j $(1 \leq j \leq k)$ in the argument t_i. Since $h \notin \{h_1, \ldots, h_k\}$, cannot be true that $\mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta} \to t_i \Leftarrow \Pi$ is $CRWL(\mathcal{D})$-provable if $Sat_{\mathcal{D}}(\Pi)$. Therefore, $Ans_{\mathcal{P}}(G)$ includes only trivial answers.

SF Assume that $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G)$. By definition, there exists a substitution $\hat{\theta}$ such that $\hat{\theta} =_{\setminus evar(G)} \theta$ and $\Pi \models_{\mathcal{D}} S\hat{\theta}$. Let $\mu \in Sol_{\mathcal{D}}(\Pi)$. Then $\hat{\theta}\mu \in Sol_{\mathcal{D}}(S)$. Since $Solve^{\mathcal{D}}(S, \chi) = fail$, we have $Sol_{\mathcal{D}}(S) = \emptyset$ (due to the requirements of the constraint solver). Hence, we also have $Sol_{\mathcal{D}}(S\hat{\theta}) = \emptyset$ and then $Sol_{\mathcal{D}}(\Pi) = \emptyset$ (it implies that $Unsat_{\mathcal{D}}(\Pi)$). Therefore, $Ans_{\mathcal{P}}(G)$ includes only trivial answers.

In both cases, $Ans_{\mathcal{P}}(G)$ includes only trivial answers, and then, $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G)$ if $Unsat_{\mathcal{D}}(\Pi)$. We prove that $Sol_{\mathcal{P}}(G) = \emptyset$: we know that $\mu \in Sol_{\mathcal{P}}(G) \Leftrightarrow (\emptyset \ \square \ \mu) \in Ans_{\mathcal{P}}(G)$. Obviously, $Sol_{\mathcal{D}}(\emptyset) = Val_{\perp}(\mathcal{D})$ and we have $Sat_{\mathcal{D}}(\emptyset)$. So, $(\emptyset \ \square \ \mu) \notin Ans_{\mathcal{P}}(G)$ because is a non-trivial answer and then $\mu \notin Sol_{\mathcal{P}}(G)$. Therefore, $Sol_{\mathcal{P}}(G) = \emptyset$.

The following soundness result follows easily from the *Correctness Lemma* and ensures that computed answers for a goal G are indeed correct answers of G.

Theorem 2 (Soundness). *If $G \vdash^* \|_{i=1}^{k} G_i$ is a concurrent derivation from G of a finite number k of goals G_i, for each $G_i \equiv \square \ \square \, S_i \, \square \, \sigma_i$ a solved goal, $S_i \, \square \, \sigma_i$ is an answer of the initial goal G. Formally, $Sol_{\mathcal{D}}(G_i) \subseteq Sol_{\mathcal{P}}(G)$.*

Completeness is based on the following idea: whenever $\Pi \, \square \, \theta \in Ans_{\mathcal{P}}(G)$ and G is not yet solved, there are finitely many local choices for a first concurrent computation step $G \vdash G_j$ $(1 \leq j \leq l)$ so that the new goals G_j are "closer to be solved" and "cover all the solutions of $\Pi \, \square \, \theta$". This idea is made precise in the next lemma, which relies on a (rather technical) *well-founded progress ordering* $\rhd$ for admissible goals, which combines similar techniques used in [5] to prove completeness of lazy narrowing calculi for *FLP* languages and for *CFLP* languages.

Lemma 2 (Progress Lemma). *Assume an admissible goal G not in solved form, and a witnessed non-trivial answer $\mathcal{M} : \Pi \, \square \, \theta \in Ans_{\mathcal{P}}(G)$. Then:*

(1) There is some concurrent goal transformation rule applicable to G.
(2) For any concurrent goal transformation rule **R** *applicable to G, there exist l goals G_j with witnessed non-trivial answers $\mathcal{M}_j : \Pi_j \square \theta_j \in Ans_{\mathcal{P}}(G_j)$ $(1 \leq j \leq l)$ such that:*

- $G \Vdash_{\mathbf{R}} \|_{j=1}^{l} G_j$,
- $Sol_{\mathcal{D}}(\Pi \square \theta) \subseteq \bigcup_{j=1}^{l} Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi_j \square \theta_j)$,
- $(G, \Pi, \theta, \mathcal{M}) \rhd (G_j, \Pi_j, \theta_j, \mathcal{M}_j)$ *for each $1 \leq j \leq l$, where $\rhd$ is a combination of well-founded progress orderings.*

Proof. (1) If $G \equiv \exists \overline{U}.\ P \square C \square S \square \sigma$ is an admissible goal not in solved form, then P or C is not empty. We will proceed by assuming gradually that no rule, except one (namely **EL**), is applicable to G, and then we will conclude that this remaining rule **EL** must be applicable. Note that failure rules cannot be applicable because otherwise G would have no answer, due to the *Correctness Lemma*. Assume that **AC** is not applicable. Then, C must be empty and the goal has the form $G \equiv \exists \overline{U}.\ P \square \square S \square \sigma$ with P not empty. Now assume that **SS**, **IM**, **PF**, **DT**, **FV** are not applicable on suspensions and **CSS**, **DI**, **DR**, **DP**, **DN**, **RRA** are not applicable on demanded productions. Then it must be the case that all the productions in P are of one of the two following forms:

1. $h\overline{e}_m \to R$ or $f\overline{e}_n\overline{a}_k \to R$ $(k \geq 0)$ or $p\overline{e}_n \to R$ or $F\overline{a}_k \to R$ $(k > 0)$, where $h\overline{e}_m$ is a rigid and passive expression but not a pattern and in all cases R is a produced but not demanded variable, in particular $R \notin dvar_{\mathcal{D}}(S)$.
2. $R'\overline{a}_k \to R$ $(k > 0)$ or $< e, \underline{case}(\tau, X, op, [\mathcal{T}_1, \ldots, \mathcal{T}_k]) > \to R$ with $e|_{pos(\tau,X)} = R'$, and both R and R' are produced and demanded variables (due to the admissibility condition **DT** [5]).

However, there cannot appear productions in P of the form (2) (otherwise, we could find a production of the form (2) and other suspension $e \to R'$ of the form (1) with R' a produced and demanded variable. Contradiction). So, all the production in P must be of the form (1) mentioned above. Consider the set χ of such R's, that is, $\chi =_{def} pvar(G)$. If the rule **CS** is not applicable, S must be in χ-solved form. But then, due to the fact that $\chi \cap dvar_{\mathcal{D}}(S) = \emptyset$ and the requirement (a) of constraint solvers, we conclude $\chi \cap var(S) = \emptyset$. Choose now some $R \in \chi$ minimal in the $\gg_P^+$ relation (such minimal elements do exist, due to the finite number of variables occurring in G and the property **NC** of admissible goals). Such R cannot appear neither in any other production in P nor in the substitution σ of the goal by the admissibility condition **SL** and then verifies $R \notin var(P \square C \square S \square \sigma)$. Therefore, the rule **EL** can be applied to the production where R appears.

(2) In each of the cases below, G' and G_i are the goals obtained by application of the corresponding transformation.

SS Let$\Pi \square \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \square C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (t \to X)\hat{\theta} \Leftarrow$

Π. Due to the *Approximation Property*, $\Pi \models_{\mathcal{D}} t\hat{\theta} \sqsupseteq X\hat{\theta}$. If we consider $\hat{\theta}' = \sigma_0\hat{\theta}$, it holds that $\hat{\theta}' =_{\setminus\{X\}} \hat{\theta}$, $\hat{\theta}' \sqsupseteq \hat{\theta}$ and $\sigma_0\hat{\theta}' = \hat{\theta}'$. Then, it is possible to lift $\hat{\theta}$ to obtain $\hat{\theta}' \sqsupseteq \hat{\theta}$ such that $\hat{\theta}' =_{\setminus\{X\}} \hat{\theta}$ and $\sigma_0\hat{\theta}' \sqsupseteq \hat{\theta}$. It follows that $\Pi \models_{\mathcal{D}} S\sigma_0\hat{\theta}'$ (if X occurs in S, $X\sigma_0\hat{\theta}' \equiv t\hat{\theta}' \equiv t\hat{\theta}$ because $X \notin var(t)$ and we know that $\sigma_0\hat{\theta}' \sqsupseteq \hat{\theta}$), $Z\hat{\theta}' \equiv s\hat{\theta}'$ for each $Z \mapsto s \in \sigma$ (the variable X does not occur in σ because is a produced variable) and $((P \;\square\; C)\hat{\theta} \Leftarrow \Pi) \succcurlyeq_{\mathcal{D}} ((P \;\square\; C)\sigma_0\hat{\theta}' \Leftarrow \Pi)$. Using the *Entailment Property*, we also have $\mathcal{P} \vdash_{\mathcal{D}} (P \;\square\; C)\sigma_0\hat{\theta}' \Leftarrow \Pi$. Hence, if we take $\hat{\theta}' =_{\setminus evar(G')} \theta'$ then $\Pi \;\square\; \theta' \in Ans_{\mathcal{P}}(G')$. Clearly, $Sol_{\mathcal{D}}(\Pi \;\square\; \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \;\square\; \theta')$.

IM Let $\Pi \;\square\; \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \;\square\; C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (h\overline{e}_m \to X)\hat{\theta} \Leftarrow \Pi$. Since $X \in dvar_{\mathcal{D}}(P \;\square\; S)$, using the *Demand Lemma* we have $\hat{\theta}(X) \neq \bot$. Furthermore, since by hypothesis $\Pi \;\square\; \theta$ is a non-trivial answer for G, we have that $Sat_{\mathcal{D}}(\Pi)$. Therefore, a proof tree $\mathcal{T}$ for $\mathcal{P} \vdash_{\mathcal{D}} (h\overline{e}_m \to X)\hat{\theta} \Leftarrow \Pi$ has the form $\mathcal{T} \equiv \mathbf{R}\ ((h\overline{e}_m \to X)\hat{\theta} \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_m])$ with $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta} \to t_i \Leftarrow \Pi$ for each $1 \leq i \leq m$. If $\hat{\theta}(X)$ is of the form $h\overline{t}_m$ then $\mathbf{R}$ is the rule **DC** and if $\hat{\theta}(X)$ is a variable, then $\Pi \models_{\mathcal{D}} h\overline{t}_m \sqsupseteq \hat{\theta}(X)$ and $\mathbf{R}$ is the rule **IR**. We define $\hat{\theta}'(X_i) = t_i$ for all $1 \leq i \leq m$ and $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for all $Y \notin \{\overline{X}_m\}$. It holds that $\sigma_0\hat{\theta}' =_{\setminus\{\overline{X}_m\}} \hat{\theta}$ if $\mathbf{R}$ is **DC** and $\Pi \models_{\mathcal{D}} \sigma_0\hat{\theta}' \sqsupseteq_{\setminus\{\overline{X}_m\}} \hat{\theta}$ if $\mathbf{R}$ is **IR**. In both cases, since $\overline{X}_m$ are new variables and X is a produced variable, it implies that $\Pi \models_{\mathcal{D}} S\sigma_0\hat{\theta}'$, $Z\hat{\theta}' \equiv s\hat{\theta}'$ for each $Z \mapsto s \in \sigma$ and $(e_i\hat{\theta} \to t_i \Leftarrow \Pi) \succcurlyeq_{\mathcal{D}} (e_i\sigma_0\hat{\theta}' \to t_i \Leftarrow \Pi)$ for each $1 \leq i \leq m$. Using the *Entailment Property*, $\mathcal{T}'_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\sigma_0\hat{\theta}' \to t_i \Leftarrow \Pi$ $(1 \leq i \leq m)$. Since $\hat{\theta}'(\sigma_0(X_i)) = t_i$, we have $\mathcal{T}'_i : \mathcal{P} \vdash_{\mathcal{D}} (e_i \to X_i)\sigma_0\hat{\theta}' \Leftarrow \Pi$ for each $1 \leq i \leq m$. Furthermore, we also have that $\overline{\mathcal{T}'} : \mathcal{P} \vdash_{\mathcal{D}} (P \;\square\; C)\sigma_0\hat{\theta}' \Leftarrow \Pi$ applying again the *Entailment Property*. Hence, if we take $\theta' =_{\setminus evar(G')} \hat{\theta}'$ then $\Pi \;\square\; \theta' \in Ans_{\mathcal{P}}(G')$. Clearly, $Sol_{\mathcal{D}}(\Pi \;\square\; \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \;\square\; \theta')$.

EL Let $\Pi \;\square\; \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \;\square\; C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e \to X)\hat{\theta} \Leftarrow \Pi$. Therefore, we have that $\Pi \;\square\; \theta \in Ans_{\mathcal{P}}(G')$ and $Sol_{\mathcal{D}}(\Pi \;\square\; \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \;\square\; \theta)$ because $X \notin var(P \;\square\; C \;\square\; S \;\square\; \sigma)$.

PF Let $\Pi \;\square\; \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \;\square\; C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (p\overline{e}_n \to X)\hat{\theta} \Leftarrow \Pi$. Since $X \in dvar_{\mathcal{D}}(P \;\square\; S)$, by the *Demand Lemma* we know that $\hat{\theta}(X) \neq \bot$. Furthermore, we have $Sat_{\mathcal{D}}(\Pi)$ because $\Pi \;\square\; \theta$ is a non-trivial answer. Furthermore, $\mathcal{T} \equiv \mathbf{PF}\ ((p\overline{e}_n \to X)\hat{\theta} \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n])$ with $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta} \to t'_i \Leftarrow \Pi$ for each $1 \leq i \leq n$ and $\Pi \models_{\mathcal{D}} p\overline{t'}_n \to! \hat{\theta}(X)$. For each $1 \leq i \leq n$, if $e_i \in Pat_{\bot}(\mathcal{U})$ then $\Pi \models_{\mathcal{D}} t'_i \sqsubseteq e_i\hat{\theta}$ by the *Approximation Property*. We define $\hat{\theta}'(X_j) = t'_j$ for each $1 \leq j \leq q$ y $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for all $Y \notin \{\overline{X}_q\}$ (if $q = 0$ we take $\hat{\theta}' = \hat{\theta}$). We have proof trees $\mathcal{T}_j : \mathcal{P} \vdash_{\mathcal{D}} e_j\hat{\theta} \to t'_j \Leftarrow \Pi$ for each $1 \leq j \leq q$. Since $\overline{X}_q$

are new variables, we also have $\Pi \models_{\mathcal{D}} S\hat{\theta}', Z\hat{\theta}' \equiv s\hat{\theta}'$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta}' \Leftarrow \Pi$ and $\mathcal{T}_j : \mathcal{P} \vdash_{\mathcal{D}} (e_j \to X_j)\hat{\theta}' \Leftarrow \Pi$ for each $1 \leq j \leq q$. On the other hand, for each $1 \leq i \leq n$, if $e_i \in Pat_{\perp}(\mathcal{U})$ then we know $t_i \equiv e_i$ and $\Pi \models_{\mathcal{D}} t'_i \sqsubseteq e_i\hat{\theta}$. Hence $\Pi \models_{\mathcal{D}} t'_i \sqsubseteq t_i\hat{\theta}'$. In other case, $t'_j = \hat{\theta}'(X_j)$, $t_j \equiv X_j$ and we have $t'_j = t_j\hat{\theta}'$. Therefore, since $\hat{\theta}(X) = \hat{\theta}'(X)$ and from $\Pi \models_{\mathcal{D}} p\overline{t'}_n \to! \hat{\theta}(X)$, we also obtain $\Pi \models_{\mathcal{D}} (p\overline{t}_n \to! X)\hat{\theta}'$. Finally, if we take $\theta' =_{\setminus evar(G')} \hat{\theta}'$ then $\Pi \sqcap \theta' \in Ans_{\mathcal{P}}(G')$. Clearly, $Sol_{\mathcal{D}}(\Pi \sqcap \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \sqcap \theta')$.

DT$_1$ Let $\Pi \sqcap \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}, Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (f\overline{e}_n \to X)\hat{\theta} \Leftarrow \Pi$. Since definitional trees are only used to control the computation, we have $\Pi \sqcap \theta \in Ans_{\mathcal{P}}(G')$ and $Sol_{\mathcal{D}}(\Pi \sqcap \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \sqcap \theta)$.

DT$_2$ Let $\Pi \sqcap \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}, Z\hat{\theta} \equiv t\hat{\theta}$ for each $Z \mapsto t \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (f\ \overline{e}_n\ \overline{a}_k \to X)\hat{\theta} \Leftarrow \Pi$. Since $X \in dvar_{\mathcal{D}}(P \sqcap S)$, the *Demand Lemma* ensures that $\hat{\theta}(X) \neq \perp$. Moreover, since $\Pi \sqcap \theta$ is a non-trivial answer of G, we have $Sat_{\mathcal{D}}(\Pi)$. Therefore, $\mathcal{T} \equiv \mathbf{DF}_{\mathcal{P}}\ (f\ \overline{e_n\hat{\theta}}\ \overline{a_k\hat{\theta}} \to \hat{\theta}(X) \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n, \mathcal{T}_c, \mathcal{T}_r, \mathcal{T}_s])$ using $(f\ \overline{t}_n \to r \Leftarrow P \sqcap C) \in [\mathcal{P}]_{\perp}$ and $s \in Pat_{\perp}(\mathcal{U})$, where $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta} \to t_i \Leftarrow \Pi$ $(1 \leq i \leq n)$, $\mathcal{T}_c : \mathcal{P} \vdash_{\mathcal{D}} P \sqcap C \Leftarrow \Pi$, $\mathcal{T}_r : \mathcal{P} \vdash_{\mathcal{D}} r \to s \Leftarrow \Pi$ and $\mathcal{T}_s : \mathcal{P} \vdash_{\mathcal{D}} \overline{sa_k\hat{\theta}} \to \hat{\theta}(X) \Leftarrow \Pi$. We define $\hat{\theta}'(X') = s$ and $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for all $Y \neq X'$. It follows that $\hat{\theta}' =_{\setminus \{X'\}} \hat{\theta}$ and since X' is a new variable does not occur in G, we have $\Pi \models_{\mathcal{D}} S\hat{\theta}', Z\hat{\theta}' \equiv Z\hat{\theta} \equiv t\hat{\theta} \equiv t\hat{\theta}'$ for each $Z \mapsto t \in \sigma$ and $\mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta}' \Leftarrow \Pi$. Furthermore, from the deduction of $\mathcal{T}_s$ is possible a derivation $\mathcal{P} \vdash_{\mathcal{D}} \hat{\theta}'(X')\ \overline{a_k\hat{\theta}'} \to \hat{\theta}'(X) \Leftarrow \Pi$, and then, $\mathcal{P} \vdash_{\mathcal{D}} (X'\ \overline{a}_k \to X)\hat{\theta}' \Leftarrow \Pi$. Moreover, we have $\mathcal{T}_i : \mathcal{P} \vdash_{\mathcal{D}} e_i\hat{\theta}' \to t_i \Leftarrow \Pi$ $(1 \leq i \leq n)$, $\mathcal{T}_c : \mathcal{P} \vdash_{\mathcal{D}} P \sqcap C \Leftarrow \Pi$ and $\mathcal{T}_r : \mathcal{P} \vdash_{\mathcal{D}} r \to \hat{\theta}'(X') \Leftarrow \Pi$. Since $Sat_{\mathcal{D}}(\Pi)$ and $\hat{\theta}'(X') = s \neq \perp$ (otherwise, the proof tree $\mathcal{T}_s$ is not possible in the $CRWL(\mathcal{D})$-calculus with $k > 0$) we can build $\mathcal{T}' \equiv \mathbf{DF}_{\mathcal{P}}\ (f\ \overline{e_n\hat{\theta}'} \to \hat{\theta}'(X') \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n, \mathcal{T}_c, \mathcal{T}_r])$ using $(f\ \overline{t}_n \to r \Leftarrow P \sqcap C) \in [\mathcal{P}]_{\perp}$. Therefore, we have $\mathcal{T}' : \mathcal{P} \vdash_{\mathcal{D}} (f\ \overline{e}_n \to X')\hat{\theta}' \Leftarrow \Pi$. Finally, if we have $\theta' =_{\setminus evar(G')} \hat{\theta}'$ then $\Pi \sqcap \theta' \in Ans_{\mathcal{P}}(G')$ and $Sol_{\mathcal{D}}(\Pi \sqcap \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \sqcap \theta')$.

FV Let $\Pi \sqcap \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\overline{\mathcal{T}} : \mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (F\ \overline{a}_k \to X)\hat{\theta} \Leftarrow \Pi$. Since $X \in dvar_{\mathcal{D}}(P \sqcap S)$, using the *Demand Lemma* we obtain $\hat{\theta}(X) \neq \perp$. Furthermore, F is a demanded variable and $\hat{\theta}(F) \neq \perp$. Since $k > 0$, we know that $\hat{\theta}(F) \notin \mathcal{U} \cup \mathcal{V}$ (in other case, the proof tree $\mathcal{T}$ is not possible in the $CRWL(\mathcal{D})$-calculus). Therefore, $\hat{\theta}(F)$ must be of the form $h\overline{t}_m \in Pat_{\perp}(\mathcal{U})$. We define $\hat{\theta}'(X_i) = t_i$ for each $1 \leq i \leq m$, $\hat{\theta}'(F) = h\overline{t}_m$ and $\hat{\theta}'(Y) = \hat{\theta}(Y)$ for all $Y \notin \{\overline{X}_m, F\}$. It holds that $\sigma_0\hat{\theta}' = \hat{\theta}'$ and $\sigma_0\hat{\theta}' =_{\setminus \{\overline{X}_m\}} \hat{\theta}$. Since $\overline{X}_m$ are new variables, it implies that $\Pi \models_{\mathcal{D}} S\sigma_0\hat{\theta}', Z\hat{\theta}' \equiv s\hat{\theta}'$ for each $Z \mapsto s \in \sigma$, $\hat{\theta}'(F) \equiv$

$h\overline{X_m\hat{\theta}'}$ for $F \mapsto h\overline{X}_m \in \sigma_0$ and $\overline{\mathcal{T}} : \mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\sigma_0\hat{\theta}' \Leftarrow \Pi$. Moreover, from $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (F\ \overline{a}_k \to X)\hat{\theta} \Leftarrow \Pi$ we also have $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (h\ \overline{X}_m\ \overline{a}_k \to X)\sigma_0\hat{\theta}' \Leftarrow \Pi$. Finally, we can take $\theta' =_{\setminus evar(G')} \hat{\theta}'$ and then $\Pi \sqcap \theta' \in Ans_{\mathcal{P}}(G')$. Clearly, $Sol_{\mathcal{D}}(\Pi \sqcap \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \sqcap \theta')$.

CSS Let $\Pi \sqcap \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ (the definitional tree is only used to control the computation). Therefore, we have $\Pi \sqcap \theta \in Ans_{\mathcal{P}}(G')$ and $Sol_{\mathcal{D}}(\Pi \sqcap \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \sqcap \theta)$.

DP, **DR**, **DI** Let $\Pi \sqcap \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ (the definitional tree is only used to control the computation). Using the *Demand Lemma*, $\hat{\theta}(R) \neq \bot$ ($R \in dvar_{\mathcal{D}}(P \sqcap C)$ by the admissibility conditions of G). By the structure of the definitional tree $\underline{case}$ (τ, X, op, $[\mathcal{T}_1, \ldots, \mathcal{T}_k]$) and since $e|_{pos(X,\tau)} = Y$, it follows that $\hat{\theta}(Y) = h_i\bar{t}_{m_i} \in Pat_{\bot}(\mathcal{U})$, where h_i ($1 \leq i \leq k$) is a passive symbol of arity m_i and $t_i \in Pat_{\bot}(\mathcal{U})$. Since $\Pi \sqcap \theta$ is a non-trivial answer of G, we have $Sat_{\mathcal{D}}(\Pi)$. Therefore, the $CRWL(\mathcal{D})$-deduction $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ uses in the last step the rule $DF_{\mathcal{P}}$ with a partial instantiation of a program rule in $\mathcal{P}$ of the form $h_i\bar{s}_{m_i}$ in the position $pos(X, \tau)$. Hence, we can define $\hat{\theta}'(Y_j) = t_j$ for each $1 \leq j \leq m_i$ and $\hat{\theta}'(X) = \hat{\theta}(X)$ for any $X \notin \{\overline{Y}_{m_i}\}$. It implies that $\sigma_0\hat{\theta}' =_{\setminus\{\overline{Y}_{m_i}\}} \hat{\theta}$. Since $\overline{Y}_{m_i}$ are new variables, we have $\Pi \models_{\mathcal{D}} S\sigma_0\hat{\theta}'$, $\mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\sigma_0\hat{\theta}' \Leftarrow \Pi$ and $Z\sigma_0\hat{\theta}' \equiv s\sigma_0\hat{\theta}'$ for each $Z \mapsto s \in \sigma$. Furthermore, since $Y \notin \{\overline{Y}_{m_i}\}$, we have $\hat{\theta}'(Y) = \hat{\theta}(Y) = h_i\bar{t}_{m_i}$ and $(h\overline{Y}_{m_i})\hat{\theta}' = h_i\overline{Y_{m_i}\hat{\theta}'} = h_i\bar{t}_{m_i}$. It follows that $Y\hat{\theta}' = (h\overline{Y}_{m_i})\hat{\theta}'$ para $Y \mapsto h\overline{Y}_{m_i} \in \sigma_0$. Since $e|_{pos(X,\tau)} = Y$ and $\hat{\theta}'(\sigma_0(Y)) = \hat{\theta}(Y) = h_i\bar{t}_{m_i}$, $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\sigma_0\hat{\theta}' \Leftarrow \Pi$ is provable with the same deduction that $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$. Therefore, if we take $\theta' =_{\setminus evar(G')} \hat{\theta}'$ we have $\Pi \sqcap \theta' \in Ans_{\mathcal{P}}(G')$ and clearly $Sol_{\mathcal{D}}(\Pi \sqcap \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \sqcap \theta')$.

DN Let $\Pi \sqcap \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ (the definitional tree is used only to control the computation). Using the *Demand Lemma*, $\hat{\theta}(R) \neq \bot$ ($R \in dvar_{\mathcal{D}}(P \sqcap C)$ by the admissibility conditions of G). Since $pos(X, \tau) \in Pos(e)$, we also have $pos(X, \tau) \in Pos(e\hat{\theta})$. Additionally, we have the following properties:

- $e\hat{\theta}$ and $e\hat{\theta}|_{pos(X,\tau)}$ have the same function symbol in the root that e and $e|_{pos(X,\tau)}$, respectively.
- Since $\tau \preceq e$ and $\tau \preceq \tau\hat{\theta}$, then $\tau\hat{\theta} \preceq e\hat{\theta}$ and $\tau \preceq e\hat{\theta}$ with τ in the root of the definitional tree $\underline{case}$ (τ, X, op, $[\mathcal{T}_1, \ldots, \mathcal{T}_k]$).

Using the *Splitting Property*, $\mathcal{P} \vdash_{\mathcal{D}} e\hat{\theta}|_{pos(X,\tau)} \to s \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} e\hat{\theta}[s]_{pos(X,\tau)} \to \hat{\theta}(R) \Leftarrow \Pi$ for $s \in Pat_{\bot}(\mathcal{U})$ ($s \neq \bot$). Hence, we can define $\hat{\theta}'(R') = s$ and

RULE	$\mid \mathcal{M} \mid$	$\mathcal{W}\ (G, \Pi, \theta, \mathcal{M})$	$\mathcal{D}\ (G)$	$\mid G \mid_0 = (\mathcal{W}\ (G, \Pi, \theta, \mathcal{M}), \mathcal{D}(G))$	$\mid G \mid_1$	$\mid G \mid_2$	$\mid G \mid_3$	$\mid S \mid$
SS	$\succeq_{mul}$	$\succeq_{mul}$	$=_{mul}$	$\succeq_{lex}$	$\geqslant_{\mathbb{N}}$	$\geqslant_{\mathbb{N}}$	$>_{\mathbb{N}}$	
IM	$\succeq_{mul}$	$\succeq_{mul}$	$=_{mul}$	$\succeq_{lex}$	$\geqslant_{\mathbb{N}}$	$>_{\mathbb{N}}$		
EL	$\succeq_{mul}$	$\succeq_{mul}$	$=_{mul}$	$\succeq_{lex}$	$\geqslant_{\mathbb{N}}$	$\geqslant_{\mathbb{N}}$	$>_{\mathbb{N}}$	
PF	$\succ_{mul}$	$\succ_{mul}$		$>_{lex}$				
DT	$\succeq_{mul}$	$\succ_{mul}$		$>_{lex}$				
FV	$\succeq_{mul}$	$\succeq_{mul}$	$=_{mul}$	$\succeq_{lex}$	$>_{\mathbb{N}}$			
CSS	$\succeq_{mul}$	$\succeq_{mul}$	$\succ_{mul}$	$>_{lex}$				
DP	$\succeq_{mul}$	$\succeq_{mul}$	$\succ_{mul}$	$>_{lex}$				
DR	$\succeq_{mul}$	$\succeq_{mul}$	$\succ_{mul}$	$>_{lex}$				
DI	$\succeq_{mul}$	$\succeq_{mul}$	$\succ_{mul}$	$>_{lex}$				
DN	$\succ_{mul}$	$\succ_{mul}$		$>_{lex}$				
RRA	$\succ_{mul}$	$\succ_{mul}$		$>_{lex}$				
CS	$\succeq_{mul}$	$\succeq_{mul}$	$=_{mul}$	$\succeq_{lex}$	$\geqslant_{\mathbb{N}}$	$\geqslant_{\mathbb{N}}$	$\geqslant_{\mathbb{N}}$	$>_{\mathbb{N}}$
AC	$\succ_{mul}$	$\succ_{mul}$		$>_{lex}$				

Fig. 5. progress ordering $(G, \Pi, \theta, \mathcal{M}) \rhd (G_j, \Pi_j, \theta_j, \mathcal{M}_j)$

$\hat{\theta'}(X) = \hat{\theta}(X)$ for any $X \neq R'$. Since R' is a new variable, we have proofs for $\mathcal{P} \vdash_{\mathcal{D}} (e|_{pos(X,\tau)} \to R')\hat{\theta'} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e[R']_{pos(X,\tau)} \to R)\hat{\theta'} \Leftarrow \Pi$. Furthermore, $\Pi \models_{\mathcal{D}} S\hat{\theta'}, Z\hat{\theta'} \equiv s\hat{\theta'}$ for each $Z \mapsto s \in \sigma$ and $\mathcal{P} \vdash_{\mathcal{D}} (P \ \square \ C)\hat{\theta'} \Leftarrow \Pi$. Therefore, if we take $\theta' =_{\setminus evar(G')} \hat{\theta'}$ we have $\Pi \ \square \ \theta' \in Ans_{\mathcal{P}}(G')$ and $Sol_{\mathcal{D}}(\Pi \ \square \ \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \ \square \ \theta')$.

RRA Let $\Pi \ \square \ \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $Z\hat{\theta} \equiv s\hat{\theta}$ for each $Z \mapsto s \in \sigma$, $\mathcal{P} \vdash_{\mathcal{D}} (P \ \square \ C)\hat{\theta} \Leftarrow \Pi$ and $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ (the definitional tree is used only to control the computation). Using the *Demand Lemma*, $\hat{\theta}(R) \neq \bot$ ($R \in dvar_{\mathcal{D}}(P \ \square \ C)$ by the admissibility conditions of G). We can assume that τ is of the form $f\bar{t}_n$ and then $e = \tau\sigma_0 = f\overline{t_n\sigma_0}$ and $e\hat{\theta} = f\overline{t_n\sigma_0\hat{\theta}}$. Since $\Pi \ \square \ \theta$ is a non-trivial answer, we have $Sat_{\mathcal{D}}(\Pi)$. Therefore, $\mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ uses the $CRWL(\mathcal{D})$-rule $_{\mathcal{P}}$ with a partial instantiation $(\tau \to r_i \Leftarrow P_i \ \square \ C_i)\mu$ of a program rule $(\tau \to r_i \Leftarrow P_i \ \square \ C_i) \in \mathcal{P}$ (for $1 \leq i \leq k$), where $\mu \in Sub_{\bot}(\mathcal{U})$ with $dom(\mu) \subseteq var(\tau \to r_i \Leftarrow P_i \ \square \ C_i)$. Therefore, the definitional tree of the deduction $\mathcal{T} : \mathcal{P} \vdash_{\mathcal{D}} (e \to R)\hat{\theta} \Leftarrow \Pi$ must be of the form, $\mathcal{T} \equiv \mathbf{DF}_{\mathcal{P}}\ ((e \to R)\hat{\theta} \Leftarrow \Pi, [\mathcal{T}_1, \ldots, \mathcal{T}_n, \mathcal{T}_c, \mathcal{T}_r])$ where $\mathcal{T}_j : \mathcal{P} \vdash_{\mathcal{D}} t_j\sigma_0\hat{\theta} \to t_j\mu \Leftarrow \Pi$ $(1 \leq j \leq n)$, $\mathcal{T}_c : \mathcal{P} \vdash_{\mathcal{D}} (P_i \ \square \ C_i)\mu \Leftarrow \Pi$ and $\mathcal{T}_r : \mathcal{P} \vdash_{\mathcal{D}} r_i\mu \to \hat{\theta}(R) \Leftarrow \Pi$. Hence, we can define $\hat{\theta'}$ as follows:

- For any variable $X \in var(\tau \to r_i \Leftarrow P_i \ \square \ C_i) \setminus dom_c(\sigma_0)$, $\hat{\theta'}(X) =_{def} \mu(X)$. In particular, $\hat{\theta'}(R_j) = \mu(R_j)$ for each $R_j \in dom_f(\sigma_0)$.
- For any variable $Y \in dom_c(\sigma_0)$, $\hat{\theta'}(Y) =_{def} \hat{\theta}(\sigma_0(Y))$.
- For any variable $Z \notin var(\tau \to r_i \Leftarrow P_i \ \square \ C_i)$, $\hat{\theta'}(Z) =_{def} \hat{\theta}(Z)$.

Then $\hat{\theta} =_{\setminus evar(G')} \hat{\theta'}$. Furthermore:

- For any $R_j \in dom_f(\sigma_0)$ $(1 \leq j \leq m)$, $\mathcal{T}'_j : \mathcal{P} \vdash_{\mathcal{D}} \hat{\theta'}(\sigma_f(R_j)) \to \hat{\theta'}(R_j)$ $(= \mu(R_j)) \Leftarrow \Pi$ is deducible with a proof which is extracted from a deduction $\mathcal{T}_j : \mathcal{P} \vdash_{\mathcal{D}} t_j\sigma_0\hat{\theta} \to t_j\mu \Leftarrow \Pi$ $(1 \leq j \leq n)$ in $\mathcal{T}$.
- Using the *Entailment Property* and since $\Pi \models_{\mathcal{D}} r_i\sigma_c\hat{\theta'} \sqsupseteq r_i\mu$, $\hat{\theta'}(R) = \hat{\theta}(R)$ and $\mathcal{T}$ have a deduction $\mathcal{T}_r : \mathcal{P} \vdash_{\mathcal{D}} r_i\mu \to \hat{\theta}(R) \Leftarrow \Pi$, it follows that $\mathcal{P} \vdash_{\mathcal{D}} r_i\sigma_c\hat{\theta'} \to \hat{\theta'}(R) \Leftarrow \Pi$ is also deducible. We prove $\Pi \models_{\mathcal{D}} \hat{\theta'}(\sigma_c(X)) \sqsupseteq \mu(X)$ for each $X \in var(r_i)$ (it implies that $\Pi \models_{\mathcal{D}} r_i\sigma_c\hat{\theta'} \sqsupseteq r_i\mu$):
  - If $X \notin dom_c(\sigma_0)$, then $\hat{\theta'}(\sigma_c(X)) = \hat{\theta'}(X) = \mu(X)$, and $\Pi \models_{\mathcal{D}} \hat{\theta'}(\sigma_c(X)) \sqsupseteq \mu(X)$.
  - If $X \in dom_c(\sigma_0)$, by the *Approximation Property*, $\hat{\theta}(\sigma_c(X)) = \hat{\theta}(\sigma_c(X))$ and $\Pi \models_{\mathcal{D}} \hat{\theta}(\sigma_c(X)) \sqsupseteq \mu(X)$ because $\hat{\theta}(\sigma_c(X))$, $\mu(X) \in Pat_{\perp}(\mathcal{U})$ and $\mathcal{P} \vdash_{\mathcal{D}} \hat{\theta}(\sigma_c(X)) \to \mu(X) \Leftarrow \Pi$ have a derivation in $\mathcal{T}_j : \mathcal{P} \vdash_{\mathcal{D}} t_j\sigma_0\hat{\theta} \to t_j\mu \Leftarrow \Pi$ $(1 \leq j \leq n)$ of $\mathcal{T}$. Therefore, $\Pi \models_{\mathcal{D}} \hat{\theta'}(\sigma_c(X)) \sqsupseteq \mu(X)$.
- $\mathcal{P} \vdash_{\mathcal{D}} (P_i \sqsubset C_i)\sigma_c\hat{\theta'} \Leftarrow \Pi$ is also $CRWL(\mathcal{D})$-deducible, because $(P_i \sqsubset C_i)\sigma_c\hat{\theta'} \sqsupseteq (P_i \sqsubset C_i)\mu$ and in $\mathcal{T}_c$ we have deductions for $\mathcal{T}_c : \mathcal{P} \vdash_{\mathcal{D}} (P_i \sqsubset C_i)\mu \Leftarrow \Pi$.
- By the admissibility conditions of G and the definition of $\hat{\theta'}$, we also have that $\Pi \models_{\mathcal{D}} S\hat{\theta'}$, $Z\hat{\theta'} \equiv Z\hat{\theta} \equiv s\hat{\theta} \equiv s\hat{\theta'}$ for each $Z \mapsto s \in \sigma$ and $\mathcal{P} \vdash_{\mathcal{D}} (P \sqsubset C)\hat{\theta'} \Leftarrow \Pi$.

Therefore, if we take $\theta' =_{\setminus evar(G')} \hat{\theta'}$ we have $\Pi \sqsubset \theta' \in Ans_{\mathcal{P}}(G')$ and clearly $Sol_{\mathcal{D}}(\Pi \sqsubset \theta) \subseteq Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi \sqsubset \theta')$.

CS Let $\Pi \sqsubset \theta \in Ans_{\mathcal{P}}(G)$. There exists $\hat{\theta} =_{\setminus evar(G)} \theta$ such that $\Pi \models_{\mathcal{D}} S\hat{\theta}$, $X\hat{\theta} \equiv t\hat{\theta}$ for each $X \mapsto t \in \sigma$ and $\overline{\mathcal{T}} : \mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta} \Leftarrow \Pi$. For each $1 \leq i \leq k$, we define $\hat{\theta}_i = mgu(\hat{\theta}, \sigma_i)$ as the *most general unifier* of the substitutions $\hat{\theta}$ and σ_i (exists $\rho_i, \delta_i \in Sub_{\perp}(\mathcal{U})$ such that $\hat{\theta}_i = \hat{\theta}\rho_i$ and $\hat{\theta}_i = \sigma_i\delta_i$) and $\Pi_i = \Pi\rho_i \wedge S_i\delta_i$. We have:

- $\Pi_i \models_{\mathcal{D}} S_i\hat{\theta}_i$. Let $\mu \in Sol_{\mathcal{D}}(\Pi_i)$. By definition of Π_i, $\mu \in Sol_{\mathcal{D}}(\Pi\rho_i)$ and $\mu \in Sol_{\mathcal{D}}(S_i\delta_i)$. Hence $\rho_i\mu \in Sol_{\mathcal{D}}(\Pi)$ and $\delta_i\mu \in Sol_{\mathcal{D}}(S_i)$. Since $\Pi \models_{\mathcal{D}} S\hat{\theta}$, we have $\rho_i\mu \in Sol_{\mathcal{D}}(S\hat{\theta})$ and $\delta_i\mu \in Sol_{\mathcal{D}}(S_i)$. Hence $\hat{\theta}\rho_i\mu \in Sol_{\mathcal{D}}(S)$ and $\delta_i\mu \in Sol_{\mathcal{D}}(S_i)$. Since $\hat{\theta}_i = \hat{\theta}\rho_i$, we have $\hat{\theta}_i\mu \in Sol_{\mathcal{D}}(S)$ and $\delta_i\mu \in Sol_{\mathcal{D}}(S_i)$ $(1 \leq i \leq k)$. Using the requirement $Sol_{\mathcal{D}}(S) = \bigcup_{i=1}^{k} Sol_{\mathcal{D}}(\exists_{\setminus Var(S)}.\ S_i \sqcap \sigma_i)$ of a solver, it follows that also $\hat{\theta}_i\mu \in Sol_{\mathcal{D}}(S_i)$ and then $\mu \in Sol_{\mathcal{D}}(S_i\hat{\theta}_i)$.
- $X\hat{\theta}_i \equiv t\hat{\theta}_i$ for each $X \mapsto t \in \sigma\sigma_i$. If $X \mapsto t \in \sigma$ then $X\hat{\theta}_i \equiv X\hat{\theta}\rho_i \equiv t\hat{\theta}\rho_i \equiv t\hat{\theta}_i$. If $X \mapsto t \in \sigma_i$ then $X\hat{\theta}_i \equiv X\sigma_i\delta_i \equiv t\delta_i \equiv t\sigma_i\delta_i \equiv t\hat{\theta}_i$.
- $\overline{\mathcal{T}}_i : \mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\sigma_i\hat{\theta}_i \Leftarrow \Pi_i$. We have $((P \sqcap C)\hat{\theta} \Leftarrow \Pi) \succcurlyeq_{\mathcal{D}} ((P \sqcap C)\hat{\theta}_i \Leftarrow \Pi_i)$ (if $(e \to t) \in P$ then exists ρ_i such that $\Pi_i \models_{\mathcal{D}} e\hat{\theta}\rho_i = e\hat{\theta}_i$, $\Pi_i \models_{\mathcal{D}} t\hat{\theta}\rho_i = t\hat{\theta}_i$ and $\Pi_i \models_{\mathcal{D}} \Pi\rho_i$. Analogously for each $(p\overline{e}_n \to! t) \in C$) and $\hat{\theta}_i = \sigma_i\delta_i = (\sigma_i\sigma_i)\delta_i = \sigma_i(\sigma_i\delta_i) = \sigma_i\hat{\theta}_i$. Then, $((P \sqcap C)\hat{\theta} \Leftarrow \Pi) \succcurlyeq_{\mathcal{D}} ((P \sqcap C)\sigma_i\hat{\theta}_i \Leftarrow \Pi_i)$. Since $\overline{\mathcal{T}} : \mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\hat{\theta} \Leftarrow \Pi$, using the *Entailment Property* we also have $\overline{\mathcal{T}}_i : \mathcal{P} \vdash_{\mathcal{D}} (P \sqcap C)\sigma_i\hat{\theta}_i \Leftarrow \Pi_i$.

Now, if we take $\theta_i =_{\setminus evar(G_i)} \hat{\theta}_i$ for each $1 \leq i \leq k$, then $\Pi_i \square \theta_i \in Ans_{\mathcal{D}}(G_i)$. Finally, we prove that $Sol_{\mathcal{D}}(\Pi \square \theta) \subseteq \bigcup_{i=1}^{k} Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi_i \square \theta_i)$. Let $\mu \in Sol_{\mathcal{D}}(\Pi \square \theta)$. By definition, $\mu \in Sol_{\mathcal{D}}(\Pi)$ and $X\mu \equiv t\mu$ for each $X \mapsto t \in \theta$. Since $\Pi \models_{\mathcal{D}} S\hat{\theta}$, we have $\mu \in Sol_{\mathcal{D}}(S\hat{\theta})$ and hence $\hat{\theta}\mu \in Sol_{\mathcal{D}}(S)$. Now, using again the requirement $Sol_{\mathcal{D}}(S) = \bigcup_{i=1}^{k} Sol_{\mathcal{D}}(\exists_{\setminus Var(S)}.\ S_i \square \sigma_i)$, it follows that $\mu \in Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ S_i\hat{\theta} \ \square \ \sigma_i\hat{\theta})$ $(1 \leq i \leq k)$. By definition, there exists $\mu' =_{\setminus Var(G)} \mu$ such that $\mu' \in Sol_{\mathcal{D}}(S_i\hat{\theta})$ and $X\mu' \equiv t\mu'$ for each $X \mapsto t \in \sigma_i\hat{\theta}$. Since $\hat{\theta}_i = mgu(\hat{\theta}, \sigma_i)$ we obtain $X\mu' \equiv t\mu'$ for each $X \mapsto t \in \hat{\theta}_i$. Moreover, since $\hat{\theta}_i = \sigma_i\delta_i$, $S_i\sigma_i = S_i$ and $\Pi_i = \Pi\rho_i \wedge S_i\delta_i$, we obtain $\mu' \in Sol_{\mathcal{D}}(\Pi_i)$. It follows that $\mu \in Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi_i \square \theta_i)$ $(1 \leq i \leq k)$. Therefore, $\mu \in \bigcup_{i=1}^{k} Sol_{\mathcal{D}}(\exists_{\setminus Var(G)}.\ \Pi_i \square \theta_i)$.

AC Analogous to the case **PF**.

Finally, the table in **Fig. 5.** shows the behavior of the different $CFLP(\mathcal{D})$ transformations in order to define a *well-founded progress ordering* $\rhd$ for goals with constrained definitional trees. By reiterated application of the previous lemma, the following completeness result is easy to prove.

Theorem 3 (Completeness). *There exists a derivation $G \Vdash^* \|_{i=1}^{k} G_i$, ending with a finite number k of solved goals $G_i \equiv \square \square S_i \square \sigma_i$, that covers all the solutions of the initial answer $S \square \sigma$. Formally, $Sol_{\mathcal{P}}(G) \subseteq \bigcup_{i=1}^{k} Sol_{\mathcal{D}}(G_i)$.*