Abstract Critical Pairs and Confluence of Arbitrary Binary Relations

Rémy Haemmerlé and François Fages

Projet Contraintes – INRIA Rocquencourt – France
FirstName.LastName@inria.fr

Abstract. In a seminal paper, Huet introduced abstract properties of term rewriting systems, and the confluence analysis of terminating term rewriting systems by critical pairs computation. In this paper, we provide an abstract notion of critical pair for arbitrary binary relations and context operators. We show how this notion applies to the confluence analysis of various transition systems, ranging from classical term rewriting systems to production rules with constraints and partial control strategies, such as the Constraint Handling Rules language CHR. Interestingly, we show in all these cases that some classical critical pairs can be disregarded. The crux of these analyses is the ability to compute critical pairs between states built with general context operators, on which a bounded, not necessarily well-founded, ordering is assumed.

Dedicated to Gérard Huet on his 60th birthday.

1 Introduction

In a seminal paper [9], Huet introduced abstract properties of term rewriting systems, and the confluence analysis of terminating term rewriting systems by critical pairs computation. Since then, the notion of critical pairs obtained by superposing the left-hand sides of rewriting rules has been applied to a wide variety of rewriting systems, ranging from pure term rewriting systems (TRS), to TRS in equational theories [14], conditional TRS [7], rewriting models of concurrency [13], rewriting logic [12,4], higher-order rewriting [2] and graph rewriting [5,15]. Similarly, Knuth-Bendix like procedures [14,10,16] for completing non confluent rewriting systems into confluent rewriting systems have been generalized to these different settings.

To date however, although the notion of critical pairs has been adapted to a variety of formalisms, there is no general definition of an abstract notion of critical pairs from which the concrete definitions could be obtained as particular instances. In the categorical formulations of rewriting systems, the notion of relative pushouts [11] does provide an abstract condition for contextual equivalences but we are not aware of an abstract critical pair lemma in the categorical setting. Recently, in the framework of canonical inference [3,6], notions of criticality and completions have been developed for abstract proof systems, assuming

F. Baader (Ed.): RTA 2007, LNCS 4533, pp. 214–228, 2007.

a well-founded ordering on proofs. However, we will show that this assumption is too strong for the confluence analysis of binary relations in dense structures, and will illustrate this situation with a concrete example.

In this paper, we provide an abstract notion of critical pair for arbitrary binary relations and context operators. We show how this notion applies to the confluence analysis of term rewriting systems, conditional TRS, and production rules with constraints such as the Constraint Handling Rules language CHR [8], under both its naive semantics and its refined semantics with partial control structures [1]. The crux of these analyses is the ability to compute critical pairs between states built with general context operators, on which a bounded, not necessarily well-founded, ordering is assumed.

The next section gives some preliminary notations about binary relations and their composition. In section 3 we propose some abstract counterparts of the notions of context, context compatibility and substitution stability from [9], and prove an abstract critical pair theorem for establishing the confluence of arbitrary binary relations. In section 4 we show how our abstract definitions can be instantiated to prove the soundness of the classical notions of critical pairs in ordinary TRS and conditional TRS. In section 5 we proceed similarly to show the soundness of the classical definitions of critical pairs in CHR, respectively under its naive semantics and under its refined semantics that includes a partial control strategy stating that a rule is fired only once on the same instances [1].

Interestingly, we show in all these cases that some classical critical pairs can be disregarded. We conclude on the generality of this work, and on some perspectives for future work.

2 Preliminaries

Let E be an arbitrary set and $\rightarrow \subset E \times E$ be an arbitrary relation on E, called here *reduction*. We shall use the following notations and definitions:

- $\mathrm{i} = \{e \rightarrow e \mid e \in E\}$ is the identity relation on E ;
- $\circ$ is the composition : $\rightarrow_a \circ \rightarrow_b = \{(e_1, e_2) \mid \exists e \in E \ (e_1 \rightarrow_a e \wedge e \rightarrow_b e_2)\}$;
- $\rightarrow^{-1} = \{(e_2, e_1) \mid (e_1 \rightarrow e_2\}$ is the inverse relation of $\rightarrow$;
- $\rightarrow^0 = \mathrm{i}$ and $\rightarrow^n = \rightarrow \circ \rightarrow^{n-1}$ for $n \geq 1$;
- $\xrightarrow{*} = \cup_{i \geq 0} \rightarrow^i$, the transitive-reflexive closure of $\rightarrow$;
- $\xrightarrow{\epsilon} = \rightarrow \cup \mathrm{i}$;
- $\uparrow = \xrightarrow{*}^{-1} \circ \xrightarrow{*}$, the *common ancestor* relation ;
- $\swarrow\!\!\!\searrow = \rightarrow^{-1} \circ \rightarrow$, the *common direct ancestor* relation.
- $\downarrow = \xrightarrow{*} \circ \xrightarrow{*}^{-1}$ the *common descendent* relation ;
- $\downarrow_\epsilon = \xrightarrow{\epsilon} \circ \xrightarrow{\epsilon}^{-1}$.
- $\rightarrow$ is *noetherian* if there is no infinite sequence $e_0 \rightarrow e_1 \rightarrow \ldots$
- $\rightarrow$ is *confluent* if $\forall e_1, e_2 \in E (e_1 \uparrow e_2 \Rightarrow e_1 \downarrow e_2)$.
- $\rightarrow$ is *locally confluent* if $\forall e_1, e_2 \in E (e_1 \swarrow\!\!\!\searrow e_2 \Rightarrow e_1 \downarrow e_2)$.
- $\rightarrow$ is *strongly confluent* if $\forall e_1, e_2 \in E (e_1 \swarrow\!\!\!\searrow e_2 \Rightarrow e_1 \downarrow_\epsilon e_2)$.

Obviously, strong confluence implies confluence, and by Newman's lemma we know that a noetherian reduction relation is confluent if and only if it is locally confluent [9]. A pair (e_1, e_2) of elements in E is $\rightarrow$*-joinable* if $e_1 \downarrow e_2$, and $\rightarrow$*-strongly joinable* if $e \downarrow_\epsilon e_2$. A set of pairs will be said $\rightarrow$-joinable if all its pairs are $\rightarrow$-joinable.

3 Abstract Critical Pairs

3.1 Abstract Contexts

In all this section, a binary relation $\rightarrow \subset E \times E$ is assumed. We provide an abstract counterpart of the notions of contexts, context compatibility and substitution stability, introduced in [9] for TRS. To this end, we study families of operators on E that generalize the operations of putting a term in a context or instantiating a term by a substitution.

Definition 1 ($\rightarrow$-compatible operators). *An n-ary operator $C : E^n \rightarrow E$ is* $\rightarrow$-compatible *if $C(e_1, \ldots, e_n) \stackrel{*}{\rightarrow} C(e_1, \ldots, e_{i-1}, e', e_{i+1}, \ldots, e_n)$ whenever $e_i \rightarrow e'$ for any index i, $1 \leq i \leq n$.*

Proposition 1. *An n-ary operator C over E is $\rightarrow$-compatible if and only if $C(e_1, \ldots, e_n) \stackrel{*}{\rightarrow} C(e'_1, \ldots, e'_n)$ whenever $e_i \stackrel{*}{\rightarrow} e'_i$ for all i, $1 \leq i \leq n$.*

Proposition 2

(i) The composition of $\rightarrow$-compatible operators is $\rightarrow$-compatible.
(ii) The projection $\pi_i^n = \lambda x_1 \ldots x_n . x_i$ (with $1 \leq i \leq n$) is $\rightarrow$-compatible.
(iii) Permuting the arguments of an operator preserves its compatibility.

Proof To prove (i) let us suppose that C_1 and C_2 are two $\rightarrow$-compatible operators over E of arity n_1 and n_2 respectively. Let $n = n_1 + n_2 - 1$ and let us suppose $e_i \stackrel{*}{\rightarrow} e'_i$ for all $1 \leq i \leq n$. We have $C_1(e_i, \ldots, e_{i+n_1}) \stackrel{*}{\rightarrow} C_1(e'_i, \ldots, e'_{i+n_1})$ by the previous proposition, for any i, $1 \leq i \leq n_2$. Furthermore we have $C_2(e_1, \ldots e_{i-1}, C_1(e_i, \ldots, e_{i+n_1}), e_{i+n_1+1}, \ldots, e_{n_2}) \stackrel{*}{\rightarrow} C_2(e'_1, \ldots, e'_{i-1}, C_1(e'_i, \ldots, e'_{i+n_1}), e'_{i+n_1+1}, \ldots, e'_{n_2})$ since $\rightarrow$ and the identity relation are included in $\stackrel{*}{\rightarrow}$. Hence the composition of C_1 and C_2 is $\rightarrow$-compatible. For (ii) and (iii), the $\rightarrow$-compatibility of the projection operators and of any $\rightarrow$-compatible operator with a permutation of its arguments follows directly from the definition. □

Definition 2 ($\rightarrow$-Contexts). *A family of* $\rightarrow$-contexts *is a family of $\rightarrow$-compatible operators containing E (as constant operators) and closed by projection, composition and argument permutation. We will denote by $\mathcal{C}^n$ the set of n-ary contexts of $\mathcal{C}$, for $n \geq 0$.*

3.2 Abstract Linear Contexts

Definition 3 (Linear $\rightarrow$-contexts). *An n-ary $\rightarrow$-context C is* linear *if whenever $e_i \rightarrow e'$ then $C(e_1, \ldots, e_n) \stackrel{\epsilon}{\rightarrow} C(e_1, \ldots, e_{i-1}, e', e_{i+1}, \ldots, e_n)$.*

A linear $\rightarrow$-context is obviously $\rightarrow$-compatible. Furthermore, we have :

Proposition 3. *The composition of linear contexts is linear. The projections are linear contexts. Permuting the arguments of a context preserves its linearity.*

Now, let us denote by e^n the sequence of n repetitions of the element e.

Definition 4 (Absorbing $\rightarrow$-contexts). *An n-ary $\rightarrow$-context C is* absorbing *if there exists an index i, $1 \leq i \leq n$, such that $\forall e_0, e_1 \ldots, e_n \in E\ C(e_1, \ldots, e_n) = C(e_1, \ldots, e_{i-1}, e_0, e_{i+1}, \ldots, e_n)$.*

Definition 5 (Linear decomposition of $\rightarrow$-contexts). *A linear n-ary $\rightarrow$-context C is a* linear decomposition *of an unary $\rightarrow$-context C' if for any element $e \in E$ we have $C'(e) = C(e^n)$. A family of linear $\rightarrow$-contexts is* linear *if any linear decomposition of its unary contexts is either unary or absorbing.*

3.3 $\mathcal{C}$-Safe Pairs

A family $\mathcal{C}$ of $\rightarrow$-contexts induces a preordering relation over pairs of elements in E as follows :

Definition 6. *The* preorder induced *by a family $\mathcal{C}$ of contexts is the relation $\geq_{\mathcal{C}}$ on pairs satisfying: $(e'_1, e'_2) \geq_{\mathcal{C}} (e_1, e_2) \Leftrightarrow \exists C \in \mathcal{C}.(e'_1 = C(e_1) \wedge e'_2 = C(e_2))$.*

Proposition 4. *$(E, \geq_{\mathcal{C}})$ is a preorder.*

Proof. The reflexivity of $\geq_{\mathcal{C}}$ follows from the fact that the projection π^1_1 is in $\mathcal{C}$. The transitivity of $\geq_{\mathcal{C}}$ follows from the closure of contexts under arbitrary compositions. □

In the following, we will denote by $>_{\mathcal{C}}$ the strict preorder associated to $\geq_{\mathcal{C}}$. In this preorder, the joinability of a pair entails the joinability of all its $\geq_{\mathcal{C}}$-greater pairs :

Lemma 1. *Let $\mathcal{C}$ be a family of $\rightarrow$-contexts and (e_1, e_2) and (e'_1, e'_2) be two pairs in E such that $(e'_1, e'_2) \geq_{\mathcal{C}} (e_1, e_2)$. (i) If $e_1 \downarrow e_2$ then $e'_1 \downarrow e'_2$. Furthermore (ii), if $\mathcal{C}$ is linear and $e_1 \downarrow_\epsilon e_2$ then $e'_1 \downarrow_\epsilon e'_2$.*

Proof. Since $(e'_1, e'_2) \geq_{\mathcal{C}} (e_1, e_2)$, let $C \in \mathcal{C}$ be a context such that $e'_1 = C(e_1)$ and $e'_2 = C(e_2)$. Concerning *(i)*, as $e_1 \downarrow e_2$, there exists an e such that $e_1 \xrightarrow{*} e$ and $e_2 \xrightarrow{*} e$. Hence by proposition 1 we have $C(e_1) \xrightarrow{*} C(e)$ and $C(e_2) \xrightarrow{*} C(e)$, hence $e'_1 \downarrow e'_2$. *(ii)* is a direct consequence of the linearity of the contexts in $\mathcal{C}$. □

One can also remark that a symmetrical pair is greater than any other pair thanks to the projection operators in $\mathcal{C}$. The joinability of symmetrical pairs is thus subsumed by the joinability of non symmetrical pairs. We call $\mathcal{C}$-Safe pairs those pairs that are joinable by the $\rightarrow$-compatibility of contexts. $\mathcal{C}$-safe pairs can thus be removed from the confluence analysis of $\nwarrow\!\!\!\!\nearrow$ pairs.

Definition 7 ($\mathcal{C}$-Safe Pairs). *A* $\mathcal{C}$-safe *pair w.r.t. a context* $C \in \mathcal{C}^2$ *and two transitions* $l_1 \to r_1$ *and* $l_2 \to r_2$ *is a pair of the form* $(C_1(s_1 \dots s_m), C_2(t_1 \dots t_n))$ *such that:*

- *for all* $1 \leq i \leq m$, $s_i \in \{l_2, r_2\}$ *and for all* $1 \leq j \leq m$, $t_j \in \{l_1, r_1\}$.
- $C_1 \in \mathcal{C}^m$ *and* $C_2 \in \mathcal{C}^n$ *are linear decompositions of respectively* $\lambda x.C(l_1, x)$ *and* $\lambda x.C(x, l_2)$.

Lemma 2. *All* $\mathcal{C}$*-safe pairs are joinable.*

Proof. In a $\mathcal{C}$-safe pair as in the definition above, we trivially have $t_i \xrightarrow{*} r_2$ for all $1 \leq i \leq m$, and $s_j \xrightarrow{*} r_1$ for all $1 \leq j \leq m$. Hence, by proposition 1, $C_1(t_1, \dots, t_m) \xrightarrow{*} C_1(r_2^m)$ and $C_2(s_1, \dots, s_n) \xrightarrow{*} C_2(r_1^n)$. Since C_1 and C_2 are the respective linear decompositions of $\lambda x.C(l_1, x)$ and $\lambda x.C(x, l_2)$ for some context C, we also have $C_1(r_2^m) = C(l_1, r_2)$ and $C_2(r_1^n) = C_1(r_1, l_2)$. The pairs are thus joinable by the $\to$-compatibility of C. □

Now, by considering the linear decomposition of $\to$-contexts, we get :

Lemma 3. *Let* $\mathcal{C}$ *be a linear familly of contexts and* C *an unary context in* $\mathcal{C}$. *If* $C_n \in \mathcal{C}^n$ *is a linear decomposition of* C *then there existe* $1 \leq i \leq n$ *such that for all* $e_1, \dots, e_n \in E$, $C_n(e_1, \dots, e_n) = C'(e_i)$.

Proof. The proof is by induction on n. The base case, where $n = 1$, is trivial. In the inductive case, where $n > 1$, since the familly $\mathcal{C}$ is linear, C_n is absorbing. Hence there exist i, $1 \leq j \leq n$, and $e' \in E$ such that for all $e \in E$ $C_n(e^n) = C_n(e^{(j-1)}, e', e^{(n-j-1)})$. The context $\lambda x_1, \dots, x_{n-1}.C_n(x_1, \dots x_{j-1}, e', x_j \dots x_{n-1})$ is thus a linear decomposition of C', and the induction hypothesis concludes the proof.

Proposition 5. *Let* $\mathcal{C}$ *be a linear family of* $\to$*-contexts. Any* $\mathcal{C}$*-safe pair w.r.t. two transitions* $l_1 \to r_1$ *and* $l_2 \to r_2$ *is of the form* $(C(l_1, r_2), C(r_1, l_2))$ *for some binary* $\to$*-context* $C \in \mathcal{C}^2$.

Proof. By lemma 3, any $\mathcal{C}$-safe pair w.r.t. some context C' and transitions $l_1 \to r_1$ and $l_2 \to r_2$, is of the form $(C'(l_1, s), C'(t, l_2))$ with $s \in \{l_2, r_2\}$ and $t \in \{l_1, r_1\}$, i.e. of the form : (i) $(C'(l_1, l_2), C'(l_1, l_2))$, (ii) $(C'(l_1, l_2), C'(r_1, l_2))$, (iii) $(C'(l_1, r_2), C'(l_1, l_2))$, or (iv) $(C'(l_1, r_2), C'(r_1, l_2))$. Hence the choice $C = \lambda x_1.x_2.\pi_1^3(x_1, x_2, C'(l_1, l_2))$ for (i), $C = \lambda x_1.x_2.\pi_1^2(C'(x_1, l_2), x_2)$ for (ii), $C = \lambda x_1.x_2.\pi_2^2(x_1, C'(l_1, x_2))$ for (iii) or $C = \lambda x_1.x_2.\pi_1^3(x_1, x_2, C'(l_1, l_2))$ for (iv) respectively, ends te proof. □

Lemma 4. *If* $\mathcal{C}$ *is a linear family of* $\to$*-context, then all* $\mathcal{C}$*-safe pairs are strongly joinable.*

Proof. By the previous proposition, we know that any $\mathcal{C}$-safe pair is of the form $(C(l_1, r_2), C(r_1, l_2))$ with $l_1 \to r_1$ and $l_2 \to r_2$. Hence, by the linearity of C we have $C(l_1, r_2) \xrightarrow{\epsilon} C(r_1, r_2)$ and $C(r_1, l_2) \xrightarrow{\epsilon} C(r_1, r_2)$. □

3.4 $\mathcal{C}$-Critical Pairs

Let us recall that an ordered set $(E, \geq)$ is *lower bounded* if any element of E is greater than or equal to some minimal element of E.

Definition 8 ($\mathcal{C}$-critical pair). *A $\mathcal{C}$-critical pair is a $\geq_{\mathcal{C}}$-minimal element of $(\nwarrow\!\!\!\!\wedge\!\!\!\!\nearrow)$ that is not $\mathcal{C}$-safe.*

Theorem 1. *Let $\mathcal{C}$ be a family of $\rightarrow$-contexts (resp. family of linear $\rightarrow$-contexts) such that $(\nwarrow\!\!\!\!\wedge\!\!\!\!\nearrow)$ is lower bounded w.r.t. $\geq_{\mathcal{C}}$. $\rightarrow$ is locally confluent (resp. strongly confluent) if and only if all $\mathcal{C}$-critical pairs are joinable (resp. strongly joinable).*

Proof. For the *if* direction, let us suppose that all $\mathcal{C}$-critical pairs are joinable. We know that any pair in $(\nwarrow\!\!\!\!\wedge\!\!\!\!\nearrow)$ is either a $\mathcal{C}$-safe pair, or is comparable to a $\mathcal{C}$-critical pair. It is thus joinable in both cases, using respectively lemma 2 and lemma 1. The "only if" is trivial since $\mathcal{C}$-critical pairs are in $(\nwarrow\!\!\!\!\wedge\!\!\!\!\nearrow)$.

The proof of strong confluence is similar using lemma 4. □

This theorem can be used to establish the local (resp. strong) confluence of an arbitrary binary relation $\rightarrow$, by finding a family $\mathcal{C}$ of (linear) $\rightarrow$-context operators for which $\geq_{\mathcal{C}}$ is lower bounded on brother pairs and admits a finite set of $\geq_{\mathcal{C}}$-minimal elements. In the following sections, we illustrate this approach on several examples. We also show that some classical critical pairs are not $\mathcal{C}$-critical, and can thus be disregarded. Furthermore, one example in section 5.2 shows a concrete case where the preordering $\geq_{\mathcal{C}}$ is not well-founded but only lower bounded on brother pairs, as the theorem requires.

4 Applications to Term Rewriting Systems

4.1 Preliminaries

Let $\mathcal{T}$ be the set of *terms* (denoted by t, s, r,...) built from a countable infinite set $\mathcal{V}$ of variables (denoted by x, y, z, ...) and a countable set $\mathcal{F}$ of function symbols (denoted by f, g, h, ...) given with their arity. We will use the classical propositions and notations borrowed from [9] :

- $\mathcal{V}(t) \subset \mathcal{V}$ for the set of variables of t,
- $\mathcal{O}(t)$ for the set of occurrences of t (denoted by u, v, ...),
- t/u for the subterm of t at u,
- $t[u \leftarrow s]$ for the subterm replacement at u,
- $u.v$ for the concatenation of u and v,
- $\leq$ the prefix ordering on occurrences,
- $u|v$ to note disjoint occurrences.

A *substitution* σ is mapping from $\mathcal{V}$ to $\mathcal{T}$ with $x\sigma = x$ almost everywhere. Substitution are extended as morphisms to $\mathcal{T}$. If $\{x_1, \ldots x_n\}$ is the *domain* of σ (i.e. the set $\{x \in \mathcal{V} | \sigma(x) \neq x\}$) we will also denote σ by $[x_1 \backslash x_1\sigma, \ldots, x_n \backslash x_n\sigma]$. A *renaming* is a substitution $[x_1 \backslash y_1, \ldots, x_n \backslash y_n]$ where the y_i's are pairwise distinct variables. A *most general unifier* (mgu) for two given terms t and s is

a substitution σ satisfying (i) $t\sigma = s\sigma$ and (ii) for all subsitution σ_1 if $t\sigma_1 = s\sigma_1$ then there exists a subsitution σ_2 such that $\sigma_1 = \sigma_2 \circ \sigma$.

A *rewriting rule* is a pair $\langle l \to r \rangle$ where l and r are two first order terms such that $l \notin \mathcal{V}$ and $\mathcal{V}(r) \subset \mathcal{V}(l)$. A *term rewriting system* $\mathcal{R}$ is a set of rewriting rules. The relation $\to_{\mathcal{R}}$ is the least relation satisfying $s \to_{\mathcal{R}} t$ if there exists a position p, a rule $\langle l \to r \rangle \in \mathcal{R}$ and a substitution σ such that $s/u = l\sigma$ and $t = s[u \leftarrow r\sigma]$.

4.2 $\mathcal{C}$-critical Pairs of Ordinary Term Rewriting Systems

Let $\mathcal{R}$ be a TRS over a set of terms $\mathcal{T}$.

Definition 9. *Let $\mathcal{C}_t$ be the family of operators over $\mathcal{T}$ containing all the operators of the form $C_{y_1,\dots y_n} = \lambda s, s_1, \dots, s_n \,.\, s[y_1 \backslash s_1, \dots, y_n \backslash s_n]$ for $\{y_1, \dots, y_n\} \subset \mathcal{V}$, and closed under composition, projection and argument permutation.*

One can easily show :

Proposition 6. *$\mathcal{C}_t$ is a family of $\to_{\mathcal{R}}$-contexts.*

Proposition 7. *$\geq_{\mathcal{C}_t}$ is well-founded.*

Therefore all subsets of $(E \times E)$, and in particular $(\curlywedge\!\!\curlyvee)$, are lower bounded. Now a *classical critical pair*, between two rules $\langle l_i \to r_i \rangle$ and $\langle l_j \to r_j \rangle$ is a pair of the form $(l_i\sigma[u \leftarrow r_j\rho\sigma], r_i\sigma)$ where :

1. ρ is a renaming and σ is a substitution such that $\mathcal{V}(l_i) \cap \mathcal{V}(l_j\rho) = \emptyset$;
2. u is an occurrence of l_i such that l_i/u is not a variable;
3. σ is an mgu for l_i/u and $l_j\rho$.

Proposition 8. *Le t be an arbitrary term and y a variable. (i) $\lambda s.C_y(s,t)$ is a linear $\to_{\mathcal{R}}$-context. (ii) $\lambda s_1 \dots s_n.C_{y_1,\dots,y_n}(t[u_1 \leftarrow y_1] \dots [u_n \leftarrow y_n], s_1, \dots, s_n)$, where the u_i's are the occurrences of y in t and y_i's are pairwise distinct variables free in t, is a linear decompositions of $\lambda s.C_y(t,s)$.*

Then by tedious case analysis, one can show :

Proposition 9. *Let $\mathcal{R}$ be a rewriting system. A $\mathcal{C}_t$-critical pair of $\to_{\mathcal{R}}$ is a classical critical pair between two rules of $\mathcal{R}$.*

This proposition together with theorem 1 establishes that the local confluence of arbitrary rewriting systems can be deduced from the joinability of its classical critical pairs. However, some classical critical pairs may be not $\mathcal{C}_t$-critical, and can thus be disregarded.

Example 1. Let $\mathcal{R}$ be the following system:

$$\mathcal{R} = \{\langle a \to f(c,b) \rangle, \langle a \to g(b) \rangle, \langle f(x,x) \to g(x) \rangle, \langle b \to c \rangle\}$$

$(f(c,b), g(b))$ is a classical critical pair for $\to_{\mathcal{R}}$. However this critical pair is a $\mathcal{C}_t$-safe pair w.r.t. the transitions $f(x,x) \to g(x)$ and $b \to c$. Indeed $C_1 = \lambda s_1, s_2.f(s_1, s_2)$ is a linear decomposition of $\lambda s.C_x(g(x), s)$, $C_2 = \lambda s.C_x(s, b)$ is linear, $f(c,b) = C_1(c,b)$ and $g(b) = C_2(g(x))$ with $c \to b$ and $f(x,x) \to g(x)$.

4.3 Strong Confluence of Linear Term Rewriting Systems

A term t is *linear* if any variable appears at most once in t. A system $\mathcal{R}$ is *linear* iff for all rules $\langle l \to r\rangle \in \mathcal{R}$, l and r are linear. Let $\to^l_{\mathcal{R}}$ be the restriction of $\to_{\mathcal{R}}$ to linear terms.

Definition 10. *Let $\mathcal{C}_l$ be the family of contexts, closed by composition, argument permutation, projection and containing the operators*

$$C^{\rho}_{y_1,\ldots,y_n} = \lambda s_0,\ldots,s_n.(\phi_0(s_0)[y_1\backslash\phi_1(s_1),\ldots,y_n\backslash\phi_n(s_n)])\rho$$

where $y_1,\ldots,y_n$ are pairwise distinct variables, ρ is a renaming, $\cup_{i\in\mathbb{N}}\{\mathcal{V}_i\}$ is a partition of $\mathcal{V}$ where the $\mathcal{V}_i$'s are infinite, and for any $i \in \mathbb{N}$, ϕ_i is a one-to-one mapping between $\mathcal{V}$ and $\mathcal{V}_i$.

Let $\mathcal{C}_s$ be the family of operators containing all substitution operators, closed by projection, argument permutation and composition.

Proposition 10. *For any relation a linear TRS $\mathcal{R}$ we have :*

(i) $\mathcal{C}_l$ and $\mathcal{C}_s$ are linear families of respectively $\to^l_{\mathcal{R}}$-contexts and $\to_{\mathcal{R}}$-contexts;
(ii) the brother pairs w.r.t. $\to^l_{\mathcal{R}}$ are lower bounded w.r.t. $\geq_{\mathcal{C}_l}$;
(iii) $\mathcal{C}_l$-critical pairs of $\to^l_{\mathcal{R}}$ are classical critical pairs;
(iv) Any $\to_{\mathcal{R}}$-brother pair is greater than or equal to a $\to^l_{\mathcal{R}}$-brother pair.

The propositions *(i)*, *(ii)* and *(iii)* can be used with theorem 1 to analyze the strong confluence of $\to^l_{\mathcal{R}}$. The strong confluence analysis of any linear rewriting system follows from *(i)*, *(iv)* and lemma 1.

4.4 Conditional Term Rewriting Systems

A Conditional Term Rewriting Systems (CTRS) is a TRS in which the application of rules is controlled by some condition. In this section we focus on particular CRTS known as *join* systems [16].

A *conditional term rewriting rule* has the form $(l \to r \Leftarrow t\downarrow t')$ where $\langle l \to r\rangle$ is a classical rewriting rule and t and t' are terms. A *conditional term rewriting system*(CTRS) is a set of conditional term rewriting rules. For a given CTRS $\mathcal{R}_{\mathcal{C}}$, the relation $\to_{\mathcal{R}_{\mathcal{C}}}$ is defined inductively by the following rule :

$$\frac{u \in \mathcal{O}(s) \qquad (l \to r \Leftarrow t\downarrow t') \in \mathcal{R}_{\mathcal{C}} \qquad t\sigma\downarrow_{\mathcal{R}_{\mathcal{C}}} t'\sigma}{s[u \leftarrow l\sigma] \to_{\mathcal{R}_{\mathcal{C}}} s[u \leftarrow r\sigma]}$$

Definition 11. *A* primary critical pair *between two conditional rules $(l_i \to r_i \Leftarrow t_i\downarrow t'_i)$ and $(\langle l_j \to r_j \Leftarrow t_j\downarrow t'_j)$ is a* conditioned pair *of the form $((t_i\sigma, t'_j\rho\sigma)\downarrow (t'_i\sigma, t'_j\rho\sigma) : (l_i\sigma[u \leftarrow r_j\rho\sigma], r_i\sigma))$ where:*

- *ρ is a renaming and σ is a substitution such that $\mathcal{V}(l_i) \cap \mathcal{V}(l_j\rho) = \emptyset$;*
- *u is an occurrence of l_i such that l_i/u is not a variable;*
- *σ is an mgu for l_i/u and $l_j\rho$;*

A conditioned pair $(t{\downarrow}t' : (s, s'))$ *is joinable if* $s\sigma{\downarrow}s'\sigma$ *for any substitution* σ *such that* $t\sigma{\downarrow}t'\sigma$*.*

This definition of critical pair was proposed in [7]. Nonetheless the authors of this paper underlined that there exist noetherian non locally confluent systems for which all those critical pairs are joinable. The following example illustrates why the previous definition is not consistent with our definition of abstract critical pairs. We then introduce the definition of secondary critical pairs.

Example 2. Let $\mathcal{R}_\mathcal{C} = \{(f(x) \to g(x) \Leftarrow x{\downarrow}a), (a \to b \Leftarrow a{\downarrow}a)\}$. The family of contexts $\mathcal{C}_t$, defined in section 4.2 for ordinary TRS, is a family of contexts for any $\to_{\mathcal{R}_\mathcal{C}}$. Nonetheless, since $f(x) \not\to_{\mathcal{R}_\mathcal{C}} g(x)$, the pair $(f(b), g(a))$ is not $\mathcal{C}_t$-safe. Furthermore, as it is minimal in $(\curlywedge\!\!\searrow)$, it is $\mathcal{C}_t$-critical and should be considered.

Definition 12. *A* secondary critical pair *between two conditional rules* $(l_i \to r_i \Leftarrow t_i{\downarrow}t'_i)$ *and* $(\langle l_j \to r_j \Leftarrow t_j{\downarrow}t'_j)$ *is a pair of the form* $((t_i, t_j\rho){\downarrow}(t'_i, t'_j\rho) : (l_i[x\backslash u[p \leftarrow r_j\rho]], r_i[x\backslash u[p \leftarrow l_j\rho]])$ *where:*

- ρ *is a renaming such that and* $\mathcal{V}(l_i) \cap \mathcal{V}(l_j\rho) = \emptyset$*;*
- u *is an arbitrary term and* $p \in \mathcal{O}(u)$*;*
- $x \in \mathcal{V}(l_i) \cap \mathcal{V}(t_i{\downarrow}t'_i)$*.*

According to the definition of joinability of conditioned pairs, a conditioned pair $(t{\downarrow}t' : (s, s'))$ defines the set of pairs $\{(s\sigma, s'\sigma) \mid t\sigma{\downarrow}t'\sigma)\}$. Let us call an *instance* of $(t{\downarrow}t' : (s, s'))$, any element of this set. With a proof analogous to proposition 9, we obtain :

Proposition 11. *Let* $\mathcal{R}_\mathcal{C}$ *be a rewriting system. A* $\mathcal{C}_t$*-critical pair of* $\to_{\mathcal{R}_\mathcal{C}}$ *is an instance of a primary or secondary critical pair between two rules of* $\mathcal{R}_\mathcal{C}$*.*

This shows the soundness of deriving the local confluence of CTRS from the joinability of both primary and secondary critical pairs. For this, an effective definition of secondary critical pairs is thus worth investigating.

5 Applications to Production Rules with Constraints

Production rules are condition-action rules that transform a base of facts by adding or removing facts at each rule firing. The Constraint Handling Rules (CHR) language [8] generalizes production rules by lifting the base of ground facts to a store of constraints over uninstantiated variables, and interpreted in an arbitrary mathematical structure.

In this section, we focus on the confluence analysis of CHR rules, and show how the abstract notion of $\mathcal{C}$-critical pairs can be instantiated to analyze the confluence of CHR rules as proposed in [1]. This is shown under both the naive semantics of CHR without control strategy, and under the refined semantics of CHR that integrates partial control strategies based on the history on rule firings, captured here by context operators. Furthermore, the necessity to deal with constrained states illustrates the difficulty to define well-founded orderings, and our use of bounded orderings instead.

5.1 Preliminaries

In CHR, a language of *built-in constraints* interpreted over some structure $\mathcal{X}$ and assumed to contain the equality $=$, is distinguished from the language of user-defined *CHR constraints* formed over a different set of predicate symbols. A CHR program is a finite sequence of CHR rules, where a CHR rule is either:

- a *simplification* rule of the form:
  $H_1, \ldots, H_i$ `<=>` $G_1, \ldots, G_j \mid B_1, \ldots, B_k$
- or a *propagation* rule of the form:
  $H_1, \ldots, H_i$ `==>` $G_1, \ldots, G_j \mid B_1, \ldots, B_k$

where $i > 0$, $j \geq 0$, $k \geq 0$, $l > 0$, $H_1, \ldots, H_i$ are CHR constraints, the guards $G_1, \ldots, G_j$ are built-in constraints, and the body $B_1, \ldots, B_k$ is composed of CHR and built-in constraints (with $k > 0$ in propagation rules).

The symbol $\top$ is used to represent empty sequences. The empty guard can be omitted together with the symbol $|$. The notation $name@R$ gives a name to a CHR rule R. For the sake of simplicity, we assume without loss of generality that a variable appears at most once in the head of a rule.

Example 3. The following CHR rules [8] define an ordering constraint solver

```
reflexivity  @ X=<Y <=> X=Y | true.
antisymmetry  @ X=<Y , Y=<X <=> X=Y.
transitivity @ W=<X , Y=<Z ==> X=Y | W=<Z.
```

The first rule eliminates the `=<` constraints with equal arguments, the second replaces a double inequality by an equality, and the third adds the transitive closure constraints.

5.2 CHR Under Its Naive Semantics

The naive operational semantics of CHR does not include any control strategy. As a result, propagation rules can loop forever. This is corrected in the refined semantics presented in the next section by imposing that a rule is fired once on the same instances. We first present the confluence analysis of CHR programs under the naive semantics.

Here, a *CHR state* is a tuple $\exists \bar{x}.\langle F, E, D\rangle$ where, $\bar{x}$ is a set of variables called *anonymous variables*, F is a multiset of built-in and CHR constraints called *goal*, E is a CHR constraint store, and D is a built-in constraint store. A state is thus a conjunction of CHR and built-in constraints[1]. In the following, we work implicitly modulo the following equivalence $\equiv$ over states :

1 $\exists \bar{x}y.\langle F, E, D\rangle \equiv \exists \bar{x}z(\langle F, E, D\rangle[y \backslash z])$ with $zy \notin \bar{x}$.
2 $\exists \bar{x}\bar{y}.\langle F, E, D\rangle \equiv \exists \bar{x}\bar{y}'.\langle F, E, D'\rangle$ if $\mathcal{X} \models \exists \bar{y}.D \Leftrightarrow \exists \bar{y}'.D'$ if $(\bar{y} \cup \bar{y}') \cap \mathcal{V}(F, E) = \emptyset$

[1] Usually a CHR goal is annotated with the free variables of the query (i.e. initial goal). Here, the anonymous variables represent the variables introduced during the computation that leads to the given state.

The condition at the end of the second rule ensures that the variables $\bar{y}$ and $\bar{y}'$ are *strictly local variables*, i.e. anonymous variables appearing only in the built-in store. Given a CHR program P, the transition relation $\rightarrow$ over states of the naive operational semantics, is defined inductively as the least relation satisfying the following rules :

Solve $\exists\bar{x}.\langle C \wedge F, E, D\rangle \rightarrow \exists\bar{x}.\langle F, E, C \wedge D\rangle$ if C is a built-in constraint
Introduce $\exists\bar{x}.\langle H \wedge F, E, D\rangle \rightarrow \exists\bar{x}.\langle F, H \wedge E, D\rangle$ if H is a CHR constraint.
Simplify $\exists\bar{x}.\langle F, H' \wedge E, D\rangle \rightarrow \exists\bar{x}\bar{y}.\langle B \wedge F, E, H = H' \wedge D\rangle$
if $(H$ `<=>` $G \mid B)$ is in P renamed with fresh variables $\bar{y}$,
and $\mathcal{X} \models D \rightarrow \exists\bar{y}(H = H' \wedge G)$.
Propagate $\exists\bar{x}.\langle F, H' \wedge E, D\rangle \rightarrow \exists\bar{x}\bar{y}.\langle B \wedge F, H' \wedge E, H = H' \wedge D\rangle$
if $(H$ `==>` $G \mid B)$ is in P renamed with fresh variables $\bar{y}$,
and $\mathcal{X} \models D \rightarrow \exists\bar{y}(H = H' \wedge G)$.

where the variables appearing in triples stand for conjunctions of constraints, and $\bar{x}$ represents the set of variables appearing in the head H.

Example 4. One possible execution of the previous program is :

$\langle$`Z=<X`, `X=<Y` $\wedge$ `Y=<Z`, *true*$\rangle$	(**Introduce** $\times 2$)
$\langle$`X=<Z` $\wedge$ `Z=<X`, `X=<Y` $\wedge$ `Y=<Z`, *true*$\rangle$	(**Propagate** `transitivity`)
$\langle$*true*, `X=<Z` $\wedge$ `Z=<X` $\wedge$ `X=<Y` $\wedge$ `Y=<Z`, *true*$\rangle$	(**Introduce** $\times 2$)
$\langle$`X=Z`, `X=<Y` $\wedge$ `Y=<Z`, *true*$\rangle$	(**Simplify** `antisymmetry`)
$\langle$*true*, `X=<Y` $\wedge$ `Y=<Z`, `X=Z`$\rangle$	(**Solve**)
$\langle$`X=Y`, *true*, `X=Z`$\rangle$	(**Simplify** `antisymmetry`)
$\langle$*true*, *true*, `X=Y` $\wedge$ `X=Z`$\rangle$	(**Solve**)

Definition 13 (Quantified conjunction). *The* quantified conjunction *of two states is a binary operator $+_{\bar{y}}$, parameterized by a set $\bar{y}$ of variables, defined by:*

$$\exists\bar{x}.\langle F, E, D\rangle +_{\bar{y}} \exists\bar{x}'.\langle F', E', D'\rangle = \exists\bar{y}\bar{x}\bar{x}'.\langle F \wedge F', E \wedge E', D \wedge D'\rangle$$

where $\bar{y}$, $\bar{x}$, $\bar{x}'$ are supposed disjoint without loss of generality. Let $\mathcal{C}_h$ be the family of quantified conjunction operators.

Proposition 12. *Quantified conjunctions are $\rightarrow$-linear.*

Unlike the orders defined in the previous section for first-order terms, the pre-order $\geq_{\mathcal{C}_h}$ may be not well-founded. This is the case when logical implication in $\mathcal{X}$ is not well-founded. For example if $\mathcal{X}$ is the constraint system $(\mathbb{N}, \leq)$, the chain $p_1 >_{\mathcal{C}_h} p_2 >_{\mathcal{C}_h} p_3 \ldots$ (where $p_i = (\langle\emptyset, \emptyset, 1 \leq x \wedge x \leq i\rangle, \langle\emptyset, \emptyset, x \leq i\rangle)$) is strictly decreasing w.r.t. $>_{\mathcal{C}_h}$.

However one can prove that the brother pairs $(\curlywedge)$ admit $\geq_{\mathcal{C}_h}$-minimal elements. For this purpose, we assume without loss of generality that no rule in P is subsumed by another one in P, since P is finit. Here we will say that a simplification rule (resp. a propagation rule) R *subsumes* another rule $(H$ `<=>` $G_1|B)$ (resp. $(H', H$ `==>` $G_1|B)$) if there exists a renaming of R of the form $(H$ `<=>` $G_2|B)$ (resp. $(H$ `==>` $G_2|B)$) such that the constraint G_2 subsumes G_1 in $\mathcal{X}$.

Proposition 13. *$(\curlywedge\!\!\searrow)$ is lower bounded with respect to $\geq_{\mathcal{C}_h}$.*

Proof. Let *min* be the mapping of valid reductions to pairs of states defined as follows :

1. $min(\exists\bar{x}.\langle C \wedge F, E, D\rangle \rightarrow \exists\bar{x}.\langle F, E, C \wedge D\rangle) = (\langle C, \emptyset, \top\rangle, \langle\emptyset, \emptyset, C\rangle)$;
2. $min(\exists\bar{x}.\langle H \wedge F, E, D\rangle \rightarrow \exists\bar{x}.\langle F, H \wedge E, D\rangle) = (\langle H, \emptyset, \top\rangle, \langle\emptyset, H, \top\rangle)$;
3. $min(\exists\bar{x}.\langle F, H' \wedge E, D\rangle \rightarrow \exists\bar{x}\bar{y}.\langle B \wedge F, E, D\rangle) = (\langle\emptyset, H', G\rangle, \langle B, \emptyset, G\rangle)$ if $(H$ `<=>` $G \mid B)$ is in P renamed with fresh variables $\bar{y}$;
4. $min(\exists\bar{x}.\langle F, H' \wedge E, D\rangle \rightarrow \exists\bar{x}\bar{y}.\langle B \wedge F, H' \wedge E, D\rangle) = (\langle\emptyset, H', G\rangle, \langle B, H', G\rangle)$ if $(H$ `==>` $G \mid B)$ is in P renamed with fresh variables $\bar{y}$

By cases on the type of the reduction, one can check that *min* is a total function, i.e. any reduction is mapped to a pair. Moreover, for any reduction $S \rightarrow S'$, $min(S \rightarrow S') = (T, T')$ defines a reduction $T \rightarrow T'$ that is smaller than or equal to any other comparable reduction.

Now we prove that any pair (S_1, S_2) in $(\curlywedge\!\!\searrow)$ is comparable to a minimal pair. Let S be a state such that $S \rightarrow S_1$ and $S \rightarrow S_2$. Let $(T_1, T_1') = min(S \rightarrow S_1)$ and $(T_2, T_2') = min(S \rightarrow S_2)$. We suppose that $T_1' \neq T_2'$, otherwise (S_1, S_2) would be symmetrical.The proof is by case on T_1 and T_2 :

- T_1 (or T_2) is of the form $\langle H, \emptyset, \top\rangle$: since $T_1' \neq T_2'$, H is not in the goal of T_2 (or T_1). Hence we deduce that $(S_1, S_2) \geq_{\mathcal{C}_h} (T_1' +_\emptyset T_2, T_1 +_\emptyset T_2')$ that is clearly minimal in $(\curlywedge\!\!\searrow)$.
- T_1 (or T_2) is of the form $\langle C, \emptyset, \top\rangle$: as in the previous case we infer that $(S_1, S_2) \geq_{\mathcal{C}_h} (T_1' +_\emptyset T_2, T_1 +_\emptyset T_2')$ that is minimal in $(\curlywedge\!\!\searrow)$.
- $T_1 = \langle\emptyset, H_1, C_1\rangle$ and $T_2 = \langle\emptyset, H_2, C_2\rangle$: let $T_1' = \langle B_1, H_1', C_1\rangle$ and $T_2' = \langle B_2, H_2', C_2\rangle$. Let $\{H_{11}, H_{12}\}$ and $\{H_{21}, H_{22}\}$ two partitions of H_1 and H_2 such that $H_{11} = H_{22}$ and $H_12 \cap H_21 = \emptyset$. Then we have $(S_1, S_2) \geq_{\mathcal{C}_h} (\langle B_1, H_1' \wedge H_{21}, G_1 \wedge G_2 \wedge H_{11} = H_{22}\rangle, \langle B_2, H_2' \wedge H_{12}, G_1 \wedge G_2 \wedge H_{11} = H_{12}\rangle)$. The latter pair is minimal in $(\curlywedge\!\!\searrow)$, as otherwise H_{11}, H_{12} and H_{21}, H_{22} would not be the biggest partitions of H_1 and H_2. □

Let $\zeta(R)$ be the constraints in the head of R that are not deleted by its application, i.e. $\zeta(R) = \mathit{true}$ if R is a simplification or $\zeta(R) = H$ if R is a propagation rule with head H. Let R be a rule with guard G, body B and head $H_1, \ldots, H_n$ and let R' be a rule renamed with fresh variables with guard G', body B' and head $H_1', \ldots, H_m'$. A *CHR critical pair* between R and R' is a pair of the form :

$$(\exists\bar{y}.\langle\zeta(R), H'_{j_{k+1}}, \ldots, H_{i_m}, B, \mathit{true}, \bar{G}\rangle, \exists\bar{y}.\langle\zeta(R'), H_{i_{k+1}}, \ldots H_{i_n}, B', \mathit{true}, \bar{G}\rangle)$$

where $\bar{G} = G \wedge G' \wedge H_{i_1} = H'_{j_1} \wedge \cdots \wedge H_{i_k} = H'_{j_k}$, while $\{i_1, \ldots, i_n\}$ and $\{j_1, \ldots, j_m\}$ are permutation of $\{1, , \ldots, n\}$ and $\{1, \ldots, m\}$ respectively, and $\bar{y}$ is the set of variables appearing in the bodies but not in the heads.

Proposition 14. *Any $\mathcal{C}_h$-critical pair is a CHR critical pair.*

This shows the soundness of analyzing the confluence of CHR programs by computing CHR critical pairs [8]. However a CHR critical pair is not necessarily a $\mathcal{C}_h$ critical pair, as shows the following :

Example 5. Let P be the CHR programe consisting into the two following rules `p,q,r <=> case(1)` and `p,q,s <=> case(2)`. P admits three critical pairs :
$p_1 = (\langle \texttt{case(1)}, \texttt{s}, true\rangle, \langle case(2), \texttt{r}, true\rangle)$
$p_2 = (\langle \texttt{case(1)}, \texttt{q} \wedge \texttt{s}, true\rangle, \langle \texttt{case(2)}, \texttt{q} \wedge \texttt{r}, true\rangle)$
$p_3 = (\langle \texttt{case(1)}, \texttt{p} \wedge \texttt{s}, true\rangle, \langle \texttt{case(2)}, \texttt{p} \wedge \texttt{s}, true\rangle)$
However, p_2 and p_3 are not $\mathcal{C}_h$-critical as $p_2 >_{\mathcal{C}_h} p_1$ and $p_3 >_{\mathcal{C}_h} p_1$.

5.3 CHR Under Its Refined Semantics

The refined operational semantics of CHR includes a partial control strategy that prevents the looping of propagation rules by restricting their firing only once on the same instances. By internalizing the necessary information in the states, one can nevertheless prove local confluence by computing critical pairs between states enriched with control tokens.

A *refined CHR state* $\exists \bar{x}.\langle\!\langle F, E, D, \mathcal{T}\rangle\!\rangle$ is composed of a naive state $\exists \bar{x}.\langle F, E, D\rangle$ together with a set $\mathcal{T}$, the *token store*, composed of tokens of the form $\mathcal{R}@C$ where C is a conjunction of constraints and $\mathcal{R}$ a rule name. $\mathcal{T}$ contains the necessary information about propagation rules, the respective constraints that can be possibly applied are contained in $\mathcal{T}$. Given a CHR program P and a CHR constraint C, the *tokenset* of an user-defined constraint C with respect to conjunction of constraints C_U is the set :

$$T(C, C_u) = \left\{ \mathcal{R}@H' \;\middle|\; \begin{array}{l} \mathcal{R}@(H \texttt{ ==> } G \mid B) \in P, C \text{ is in } H', \\ H' \text{ is a subset of } C \wedge C_u, H \text{ unifies with } H' \end{array} \right\}$$

and $T(C_1 \wedge \cdots \wedge C_n, C_u) = T(C_1, C_u) \cup \cdots \cup T(C_n, C_u)$. The refined transition relation $\rightarrow_+$ is given by the following rules, where the variables appearing in triples stand for conjunctions of constraints and $\bar{x}$ represents the set of variables appearing in the head H.

Solve
$\exists \bar{x}.\langle\!\langle C \wedge F, E, D, \mathcal{T}\rangle\!\rangle \rightarrow_+ \exists \bar{x}.\langle\!\langle F, E, C \wedge D, \mathcal{T}\rangle\!\rangle$ if C is a built-in constraint
Introduce
$\exists \bar{x}.\langle\!\langle H \wedge F, E, D, \mathcal{T}\rangle\!\rangle \rightarrow_+ \exists \bar{x}.\langle\!\langle F, H \wedge E, D, \mathcal{T} \cup T(H, E)\rangle\!\rangle$
if H is a CHR constraint.
Simplify
$\exists \bar{x}.\langle\!\langle F, H' \wedge E, D, \mathcal{T}\rangle\!\rangle \rightarrow_+ \exists \bar{x}\bar{y}.\langle\!\langle B \wedge F, E, D, \mathcal{T} \cap T(H \wedge E, \top)\rangle\!\rangle$
if $(H \texttt{ <=> } G \mid B)$ is in P renamed with fresh variables $\bar{y}$,
and $\mathcal{X} \models D \rightarrow \exists \bar{x}(H = H' \wedge G)$.
Propagate
$\exists \bar{x}.\langle\!\langle F, H' \wedge E, D, \{\mathcal{R}@H'\} \cup \mathcal{T}\rangle\!\rangle \rightarrow_+ \exists \bar{x}\bar{y}.\langle\!\langle B \wedge F, H' \wedge E, D, \mathcal{T}\rangle\!\rangle$
if $\mathcal{R}@(H \texttt{ ==> } G \mid B)$ is in P renamed with fresh variables $\bar{y}$,
and $\mathcal{X} \models D \rightarrow \exists \bar{x}(H = H' \wedge G)$.

Definition 14 (Refined state conjunction). *The* refined quantified conjunction *of two states is a binary operator $+_{\bar{y}}$, parameterized by a set $\bar{y}$ of variables and defined as:*

$$\exists \bar{x}.\langle\!\langle F, E, D, \mathcal{T} \rangle\!\rangle +_{\bar{y}} \exists \bar{x}'.\langle\!\langle F', E', D', \mathcal{T}' \rangle\!\rangle = \exists \bar{y}\bar{x}\bar{x}'.\langle\!\langle F \wedge F', E \wedge E', D \wedge D', \mathcal{T} \cup \mathcal{T}' \rangle\!\rangle$$

where $\bar{y}$, $\bar{x}$, $\bar{x}'$ are supposed disjoint without loss of generality. Let $\mathcal{C}_h^+$ be the family of refined state conjunction operators.

Definition 15. *Let R be a rule with guard G, body B and head $H_1, \ldots, H_n$ and let R' be a rule renamed with fresh variables with guard G', body B' and head $H'_1, \ldots, H'_m$. A* refined critical pair *between R and R' is a pair of the form : $(\exists \bar{y}.\langle\!\langle \zeta(R), H'_{j_{k+1}} \ldots, H_{i_m} B, true, \bar{G}, \emptyset \rangle\!\rangle, \exists \bar{y}.\langle\!\langle \zeta(R'), H_{i_{k+1}} \ldots H_{i_n}, B', true, \bar{G}, \emptyset \rangle\!\rangle)$ where $\bar{G} = G \wedge G' \wedge H_{i_1} = H'_{j_1} \wedge \cdots \wedge H_{i_k} = H'_{j_k}$, while $\{i_1, \ldots, i_n\}$ and $\{j_1, \ldots, j_m\}$ are permutation of $\{1, , \ldots, n\}$ and $\{1, \ldots, m\}$ respectively and $\bar{y}$ is the set of variables appearing in bodies but not in the heads.*

With a proof similar to the previous proposition, we get :

Proposition 15. *Any $\mathcal{C}_h^+$-critical pair is a CHR refined critical pair.*

6 Conclusion

By abstracting the notion of critical pairs from term rewriting systems to arbitrary binary relations and arbitrary context operators over some set E, an abstract critical pair theorem has been proved, and shown useful to establish the local confluence of a wide variety of transition systems. This has been illustrated by instantiating the abstract notion of contexts and critical pairs to prove the soundness of classical critical pair definitions in term rewriting systems, conditional term rewriting systems, and Constraint Handling Rules programs. In the latter case, our use of bounded orderings instead of well-founded orderings has been shown necessary to handle the constrained states of CHR transitions. Interestingly in all these cases, we have shown that some classical critical pairs could be disregarded.

An abstract notion of linear contexts and linear decomposition has been proved useful to establish these results. This could be further developed to define an abstract notion of orthogonal systems [16]. As for future work, the generalization of our approach to n-ary relations might also be worth investigating in connection to the theory of canonical inference [3,6], with well-founded ordering assumptions replaced by boundedness conditions.

Acknowledgments. We are grateful to the anonymous referees for their comments and for pointing us to [3]. This work benefited from partial support by the ANR RNTL Manifico project.

References

1. Abdennadher, S.: Operational semantics and confluence of constraint propagation rules. In: Smolka, G. (ed.) Principles and Practice of Constraint Programming - CP97. LNCS, vol. 1330, pp. 252–266. Springer, Heidelberg (1997)
2. Blanqui, F.: Termination and confluence of higher-order rewrite systems. In: Proceedings of the 11th International Conference on Rewriting Techniques and Applications, Lecture Notes in Computer Science, pp. 47–61 (2000)
3. Bonacia, M., Dershowitz, N.: Abstract canonical inference. ACM Transactions on Computaional Logic, 8(1) (2007)
4. Clavel, M., Durán, F., Eker, S., Meseguer, J.: Building equational proving tools by reflection in rewriting logic. In: Proceedings of the CafeOBJ Symposium '98. Japan Advanced Institute for Science and Technology (1998)
5. Corradini, A., Montanari, U.: An algebra of graphs and graph rewriting. In: Proceedings of the 4th International Conference on Category Theory and Computer Science, Lecture Notes in Computer Science, pp. 236–260 (1991)
6. Dershowitz, N., Kirchner, C.: Abstract canonical presentations. Journal of Theorical Computer Science 357, 53–69 (2006)
7. Dershowitz, N., Okada, M., Sivakumar, G.: Confluence of conditional rewrite systems. In: Kaplan, S., Jouannaud, J.-P. (eds.) Proceedings of the First International Workshop on Conditional Term Rewriting Systems. LNCS, vol. 308, pp. 31–44. Springer, Heidelberg (1988)
8. Frühwirth, T.: Theory and practice of constraint handling rules. Journal of Logic Programming, Special Issue on Constraint Logic Programming 37(1-3), 95–138 (1998)
9. Huet, G.: Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems. Journal of the ACM 27(4), 797–821 (1980)
10. Jouannaud, J.-P., Kirchner, H.: Completion of a set of rules modulo a set of equations. SIAM Journal of Computing 15(4), 1155–1194 (1986)
11. Leifer, J.J., Milner, R.: Deriving bisimulation congruences for reactive systems. In: Proceedings of the 11th International Conference on Concurrency Theory, pp. 243–258 (2000)
12. Meseguer, J.: Rewriting logic as a semantic framework for concurrency: a progress report. In: Proceedings of the 7th International Conference on Concurrency Theory, pp. 331–372 (1996)
13. Noll, T.: On coherence properties in term rewriting models of concurrency. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, pp. 478–493. Springer, Heidelberg (1999)
14. Peterson, G., Stickel, M.: Complete sets of reductions for some equational theories. Journal of the ACM 28(2), 233–264 (1981)
15. Raoult, J., Voisin, F.: Set-theoretic graph rewriting. In: Proceedings of the International Workshop on Graph Transformations in Computer Science, Lecture Notes in Computer Science, pp. 312–325 (1993)
16. Terese.: Term Rewriting Systems. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge (2003)