First-Order Atom Definitions Extended

Bijan Afshordel, Thomas Hillenbrand, and Christoph Weidenbach

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
D-66123 Saarbrücken
{afshorde,hillen,weidenb}@mpi-sb.mpg.de

Abstract. We extend the notion of atom definitions in first-order formulae by guards. These are conditions restricting the atom definition in a form that frequently occurs in many application areas of automated theorem proving. We give a sufficient and complete criterion for a formula to contain such a definition and provide an effective algorithm to actually retrieve the definition in an applicable form. An implementation within our prover SPASS leads to significant performance improvements in application areas where atom definitions are present.

1 Introduction

The question of when to expand and when to contract a definition is mentioned by Larry Wos as the 30th of his 33 basic research problems [6] in automated theorem proving, commented that "wise use of both definition expansion and definition contraction could sharply increase the effectiveness of various procedures that manipulate large amounts of knowledge".

As a supposition to the problem in the context of automated reasoning, techniques have to be developed to actually detect, expand and contract definitions. This paper is a contribution to the automatic detection and expansion of (atom) definitions that generalizes existing work in that a definition may be restricted by a so-called guard formula.

The conventional notion of atom definitions is logical equivalence of atom instances and replacement formulae, i.e. an expression

$$\forall x_1, \ldots, x_m \, (P(t_1, \ldots, t_n) \equiv \psi).$$

However, when we were working with our formula translator, practice revealed that this notion is too weak to successfully deal with important application domains such as set theory or software verification/analysis where typically such an equivalence holds under certain guarding conditions only.[1] In this paper we therefore work out the notion of an atom definition extended by a guard as in

$$\forall x_1, \ldots, x_m \, (\phi \supset (P(t_1, \ldots, t_n) \equiv \psi)).$$

Now in order to apply such a definition, the subformula ϕ, the so-called guard, must be valid in the context of the atom to be replaced. For this subproof, we

[1] For example, in the software context, type information already builds guard formulae.

R. Nieuwenhuis and A. Voronkov (Eds.): LPAR 2001, LNAI 2250, pp. 309–319, 2001.
© Springer-Verlag Berlin Heidelberg 2001

employ the very same techniques that already showed to be useful in the context of optimized Skolemization [2]. Furthermore, a formula may not always contain an atom definition in the nice form presented above. For this case we give a syntactic criterion for a formula to contain an atom definition and provide an effective algorithm to actually retrieve the definition in the above applicable form. We show that the algorithm is correct and complete with respect to the syntactic criterion. A similar procedure is used to derive the application context.

A typical example for an atom definition with guard in the set theory domain is the definition of function composition: If f_{AB}, read as a triple, denotes a function from the set A into the set B, and g_{BC} from B to C, then composition can be defined in a point-wise fashion as follows:

$$\forall f_{AB}, g_{BC}, x, z \\ (x \in A \wedge z \in C) \supset \\ [g_{BC} \circ f_{AB}(x) = z \equiv \exists y\,(y \in B \wedge f_{AB}(x) = y \wedge g_{BC}(y) = z)].$$

Here $(x \in A \wedge z \in C)$ is the guard that has to be checked before the atom definition of $g_{BC} \circ f_{AB}(x) = z$ can be applied.

In the context of clause-based automated theorem proving, the detection and expansion of atom definitions is a preprocessing step of the clause normal form (CNF) translation procedure. It is build into the SPASS transformation process that is organized in three stages: *formula renaming*, *optimized Skolemization* and *simplification*. In the first stage, subformulae are replaced with new predicates and the definitions thereof added, provided the replacement eventually leads to fewer clauses. This step is closely related to the automatic detection and contraction of definitions. In the optimized Skolemization stage existential quantifiers are removed in exchange for carefully introduced new functions. In the last stage domain specific simplification techniques, e.g. based on the domain size, are applied along with more general techniques like equality simplifications and expansion of (atom) definitions. This expansion step can be considered as symmetrical to formula renaming.

Thus by adding the techniques described in this paper to SPASS, we complete the picture of techniques in the CNF transformation process. Altogether we can now provide a framework within which appropriate instantiation strategies can be developed and tested.

An experimental evaluation gives strong evidence that our approach is promising. It was performed in the SET (set theory) application area of the TPTP [4], feeding the resulting clausal specifications into the SPASS theorem prover. This way we could solve about 60 extra problems that make up almost 20% of the overall first-order SET problems in the TPTP. For many of these newly solved proof tasks, no previous fully automated proof attempt with any other prover had been reported.

Now the paper is organized as follows: Section 2 introduces the relevant notions and notations. In Section 3 we develop the notion of an atom definition and show under which conditions the expansion of atom definitions preserves equivalence/satisfiability. As atom definitions do not always have the desired shape required in Section 3 the shape has often first to be extracted from a given

formula. Section 4 provides a syntactic criterion for the existence of an atom definition and an effective correct and complete algorithm to actually compute the desired shape. Section 5 demonstrates that the theory is successfully implemented in SPASS. We end the paper with a short summary and a discussion of related work, Section 6.

2 Preliminaries

We rely on the terminology described in [2]. Therefore, we only repeat the more sophisticated concepts and some notation.

We recursively construct *formulae* over atoms, the logical constants $\top$ (truth), $\bot$ (falsity) and the operators $\supset$ (implication), $\equiv$ (equivalence), $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation) and the quantifiers $\forall$ (universal), $\exists$ (existential) as usual. We sometimes omit parentheses to ease readability; especially, highest priority is given to quantification, negation, equivalence, implication, in descending order.

For the sequence $x_1, \ldots, x_n$ $(t_1, \ldots, t_n)$ we use the abbreviation $\overline{x_n}$ $(\overline{t_n})$. For convenience, we often write $\forall x_1, \ldots, x_n\, \phi$ or $\overline{\forall x_n}$ instead of $\forall x_1 \ldots \forall x_n\, \phi$ and analogously for the existential quantifier. If $Q_1, \ldots, Q_n$ denotes a sequence of quantifier symbols, then $\overline{Q_n x_n}$ is short for $Q_1 x_1 \ldots Q_n x_n$. To simplify the discussion we always assume that in a formula any variable is bound by at most one quantifier.

An *interpretation* is a triple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \nu \rangle$ where $\mathcal{D}$ is a non-empty set, namely the domain of discourse, $\mathcal{I}$ associates n-ary predicate symbols and function symbols with n-place relations and functions respectively. We use $\mathcal{M}[x/a]$ as a short form for $\langle \mathcal{D}, \mathcal{I}, \nu[x/a] \rangle$ provided $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \nu \rangle$. The *satisfies* relation $\models$ is defined as usual.

For the purpose of this paper, we extend substitutions to formulae as follows: $P(t_1, \ldots, t_n)\sigma = P(t_1\sigma, \ldots, t_n\sigma)$, $(\neg\phi)\sigma = \neg(\phi\sigma)$, $(\phi_1 \circ \phi_2)\sigma = \phi_1\sigma \circ \phi_2\sigma$ where $\circ \in \{\supset, \equiv, \wedge, \vee\}$, $(\forall x\, \phi)\sigma = \forall x\sigma\, \phi\sigma$ if $x\sigma$ is a variable and $\forall x\, \phi$ otherwise, $(\exists x\, \phi)\sigma = \exists x\sigma\, \phi\sigma$ if $x\sigma$ is a variable and $\exists x\, \phi$ otherwise. In general, the extension of substitutions to formulae is problematic, because it might accidentally turn free variables into bound variables. However, we shall only make use of this definition in very restricted contexts where the desired semantics is preserved by the application of the substitution.

A *position* is a word over the natural numbers. The set $pos(\phi)$ of positions of a given formula ϕ is defined as follows: (i) the empty word $\epsilon \in pos(\phi)$ (ii) $i.\pi \in pos(\phi)$ if $\phi = \phi_1 \circ \phi_2$ and $\pi \in pos(\phi_i)$, $i \in \{1, 2\}$ where $\circ \in \{\supset, \equiv, \wedge, \vee\}$ and (iii) $1.\pi \in pos(\phi)$ if $\phi = \neg\psi$ or $\phi = \forall x\, \psi$ or $\phi = \exists x\, \psi$ and $\pi \in pos(\psi)$. Now, if $\pi \in pos(\phi)$ we define $\phi|_\epsilon = \phi$ and $\phi|_{i.\tau} = \phi_i|_\tau$ where $\phi = \phi_1 \circ \phi_2$, $\circ \in \{\supset, \equiv, \wedge, \vee\}$ and define $\phi|_{1.\tau} = \psi|_\tau$ if $\phi = \neg\psi$ or $\phi = \forall x\, \psi$ or $\phi = \exists x\, \psi$. We write $\psi[\phi]_\pi$ for $\psi|_\pi = \phi$. With $\psi[\pi/\phi]$, where $\pi \in pos(\psi)$, we denote the formula obtained by replacing $\psi|_\pi$ with ϕ at position π in ψ. The *length* of a position π is defined by $|\epsilon| = 0$ and $|i.\tau| = 1 + |\tau|$. Let $\leq$ denote the usual prefix ordering on positions:

$\pi \leq \tau$ iff there exists a position ι with $\pi.\iota = \tau$, and $\pi < \tau$ iff $\pi \leq \tau$ and not $\tau \leq \pi$.

The polarity of a formula occurring at position π in a formula ψ is denoted by $pol(\psi, \pi)$ and is defined in the usual way: $pol(\psi, \epsilon) = 1$; $pol(\psi, \pi.i) = pol(\psi, \pi)$ if $\psi\,|_\pi$ is a conjunction, disjunction, formula starting with a quantifier or an implication with $i = 2$; $pol(\psi, \pi.i) = -pol(\psi, \pi)$ if $\psi|_\pi$ is a formula starting with a negation symbol or an implication with $i = 1$ and, finally, $pol(\psi, \pi.i) = 0$ if $\psi|_\pi$ is an equivalence.

The set of *free*, *bound* and *all* variables of a formula ϕ (term t) is defined in the usual way and denoted by the functions $free()$, $bound()$ and $vars()$, respectively.

3 Atom Definitions and Their Application

In this section we develop the notion of an atom definition and its expansion inside some other sentence. We give necessary conditions for an expansion to preserve satisfiability or even equivalence.

Definition 1 (Atom definition).
An *atom definition for a predicate* P is a sentence of the form

$$\forall x_1, \ldots, x_m \, (\varphi \supset P(t_1, \ldots, t_n) \equiv \psi)$$

where $free(P(t_1, \ldots, t_n)) = \{x_1, \ldots, x_m\}$. The formula φ is called the *guard formula* of the atom definition and ψ its *expansion formula*. We also say that $\forall x_1, \ldots, x_m \, (\varphi \supset P(t_1, \ldots, t_n) \equiv \psi)$ is an atom definition for P at position $1^m.2$.

Note that we consider the above definition with respect to the symmetry of $\equiv$. We do not restrict the shape of the expansion term. For example, the defined predicate P may occur in the expansion formula, since practice has shown that even the replacement of the atom by such an expansion formula can be of great benefit. As only sentences are considered, the condition $free(P(t_1, \ldots, t_n)) = \{x_1, \ldots, x_m\}$ already implies that $free(\psi), free(\varphi) \subseteq free(P(t_1, \ldots, t_n))$ which makes sense, because eventually we want to apply the atom definition to an atom $P(s_1, \ldots, s_n)$ occurring in some sentence ϕ_2 without creating free variables.

We want to replace the atom $P(s_1, \ldots, s_n)$ in ϕ_2 by the expansion formula of the definition. In order to do this we first have to find a matcher σ with $P(t_1, \ldots, t_n)\sigma = P(s_1, \ldots, s_n)$. Second, we have to check the validity of the guard in the context of $P(s_1, \ldots, s_n)$. In order to make this more precise, let us assume without loss of generality that $\phi_2 = Qz_{i_1} \ldots Qz_{i_j}(\varphi' \supset \phi[P(s_1, \ldots, s_n)]_\pi)$ where φ' is the already mentioned context of $P(s_1, \ldots, s_n)$. Note that if $j = 0$ and $\varphi' = \top$ the sentence represents the case where no context is explicitly available. So we can further assume that $\{z_{i_1}, \ldots, z_{i_j}\} \subseteq vars(P(s_1, \ldots, s_n))$. Then proving the validity of the guard amounts to showing $\models \forall z_1, \ldots, z_k(\varphi' \supset \varphi\sigma)$, where $\{z_1, \ldots, z_k\} = vars(P(s_1, \ldots, s_n))$. The theorem below states that in fact if all conditions we discussed so far are fulfilled, the replacement of the atom by the expansion formula preserves equivalence.

Theorem 2. Let $\phi_1 = \forall x_1, \ldots, x_m\,(\varphi \supset P(t_1, \ldots, t_n) \equiv \psi)$ be an atom definition, let $\phi_2 = Qz_{i_1} \ldots Qz_{i_j}(\varphi' \supset \phi[P(s_1, \ldots, s_n)]_\pi)$ be a sentence to which we want to apply the atom definition, let both sentences have no variable symbols in common and let σ be a substitution with $P(t_1, \ldots, t_n)\sigma = P(s_1, \ldots, s_n)$. Furthermore we assume $\models \forall z_1, \ldots, z_k(\varphi' \supset \varphi\sigma)$ where $\{z_{i_1}, \ldots, z_{i_j}\} \subseteq \{z_1, \ldots, z_k\} = \mathit{vars}(P(s_1, \ldots, s_n))$. Then

$$\mathcal{M} \models \phi_1 \wedge \phi_2 \quad \text{iff} \quad \mathcal{M} \models \phi_1 \wedge \phi_2[1^j.2.\pi/\psi\sigma]$$

Proof. Note that $\mathit{free}(\varphi') \subseteq \{z_{i_1}, \ldots, z_{i_j}\}$ as well as $\mathit{free}(\varphi\sigma) \subseteq \{z_1, \ldots, z_k\}$ and $\mathit{free}(\psi\sigma) \subseteq \{z_1, \ldots, z_k\}$, because both ϕ_1 and ϕ_2 are sentences. We only show the left-to-right part of the theorem, the other part is a simple syntactic variant of this one. So let us assume $\mathcal{M} \models \phi_1 \wedge \phi_2$ and we must show $\mathcal{M} \models \phi_2[1^j.2.\pi/\psi\sigma]$, in more detail $\mathcal{M} \models Qz_{i_1} \ldots Qz_{i_j}(\varphi' \supset \phi[\pi/\psi\sigma])$. Let us assume that $\mathcal{M}[z_{i_1}/a_1, \ldots, z_{i_j}/a_j] \models \varphi'$ for some arbitrary $a_1, \ldots, a_j$, for otherwise we are already done. Now let $\{z_{h_1}, \ldots, z_{h_l}\} = \{z_1, \ldots, z_k\} \setminus \{z_{i_1}, \ldots, z_{i_j}\}$. Then for arbitrary $a_{j+1}, \ldots, a_k$ we know that $\mathcal{M}[z_{i_1}/a_1, \ldots, z_{i_j}/a_j][z_{h_1}/a_{j+1}, \ldots, z_{h_l}/a_k] \models P(s_1, \ldots, s_n) \equiv \psi\sigma$, because $P(t_1, \ldots, t_n)\sigma = P(s_1, \ldots, s_n)$, $\models \forall z_1, \ldots, z_k(\varphi' \supset \varphi\sigma)$, $\mathcal{M} \models \phi_1$ and ϕ_1, ϕ_2 have no variables in common. Finally, by an inductive argument on the length of π we conclude $\mathcal{M}[z_{i_1}/a_1, \ldots, z_{i_j}/a_j] \models \phi[\pi/\psi'\sigma]$. □

There are several aspects that can be further discussed here. First, a formula representing an atom definition needs not be of the shape introduced in Definition 1. Furthermore, the context in the application formula may not be syntactically available in the form required by Theorem 2. A possible answer to both questions is contained in Section 4, where techniques are discussed that actually extract atom definitions. The very same techniques can also be applied to retrieve potential contexts of an atom to be replaced in a formula.

Second, the validity of the guard formula $\models \forall z_1, \ldots, z_k(\varphi' \supset \varphi\sigma)$ is undecidable in general. We already explored this kind of problem when we introduced optimized Skolemization [2]. In practice, the answer is to apply an inference procedure for this proof attempt that is guaranteed to terminate but still valuable. In SPASS we employ an implementation of depth-bounded unit resolution for this purpose [5].

Third, if the predicate P does not occur in the expansion formula ψ and if all occurrences of P in other formulae can be replaced, then the atom definition can be deleted, preserving satisfiability. Of course, this does only apply to predicates that are freely interpreted. For example, equality cannot be eliminated this way.[2]

4 Contained Atom Definitions

Atom definitions considered so far have been required to be presented in the syntactical form of a universally quantified conditional equivalence. In order to increase the number of situations where our concept is applicable, we now

[2] Except all equality axioms are explicitly added before the expansion is started.

develop a notion of atom definitions being contained in other formulae such that the superformulae can, preserving equivalence, effectively be transformed into the syntactical form of Definition 1. This may be looked upon as extracting definitions from formulae. We start with the following technical term:

Definition 3 (Contain disjunctively). Let ϕ be a formula containing a subformula φ at a position π. If for all $\nu \in pos(\phi)$ with $\nu < \pi$, $\phi|_\nu$ is
(i) a disjunction or implication with $pol(\phi, \nu) = 1$, or
(ii) a conjunction with $pol(\phi, \nu) = -1$, or
(iii) a quantification or a negation,
then we say that ϕ *contains* φ *disjunctively at* π.

The essence of this notion is that, the three conditions being fulfilled, the formula ϕ can effectively be transformed into an equivalent one of the form $\overline{Q_k x_k}(\psi \vee \varphi)$ or $\overline{Q_k x_k}(\psi \vee \neg\varphi)$, depending on the polarity $pol(\phi, \pi)$. This will be useful within our extraction algorithm; but to result in atom definitions, further conditions are necessary.

Definition 4 (Contain atom definition). Let ϕ be a formula with a subformula $P(t_1, \ldots, t_n) \equiv \psi$ or $\psi \equiv P(t_1, \ldots, t_n)$ at position π. Then ϕ is said to *contain an atom definition for the predicate* P *at* π if
(i) ϕ contains disjunctively the subformula $\phi|_\pi$ with $pol(\phi, \pi) = 1$,
(ii) each free variable in $P(t_1, \ldots, t_n)$ is bound by a universal quantifier with polarity 1 or by an existential quantifier with polarity -1,
(iii) $free(\psi) \subseteq free(P(t_1, \ldots, t_n))$ and
(iv) for all positions ξ, ν with $\xi < \nu \leq \pi$ we have: if $\phi|_\nu$ is a quantification binding a free variable in $P(t_1, \ldots, t_n)$, and $\phi|_\xi$ is a quantification, then the latter also binds a free variable in $P(t_1, \ldots, t_n)$.

Now the algorithm to extract atom definitions needs to be formulated. It will be specified as stratified application of extraction rules that perform a transformation of formulae. The rules are organized in three groups.

Definition 5 (Extraction rules). The list in Table 1 contains the *extraction rules* as set of transformations on formulae and positions. Note that for each of the rules in groups *(R2)* and *(R3)*, due to the symmetry of the operators $\vee$ and $\equiv$, copies with corresponding permutations of left-hand sides have been omitted to ease readability.

The application of extraction rules preserves a number of important properties that are listed in the subsequent lemma. The proof is simply by inspection of the transformations.

Lemma 6. Consider an extraction step $(\phi, \pi) \to (\phi', \pi')$.
(i) Then ϕ is logically equivalent to ϕ'.
(ii) The polarities $pol(\phi, \pi)$ and $pol(\phi', \pi')$ are the same.
(iii) If ϕ contains $\phi|_\pi$ disjunctively at π, then so does ϕ' with $\phi'|_{\pi'}$ at π'.
(iv) If ϕ contains an atom definition for a predicate P at π, so does ϕ' at π'.

Table 1. Extraction Rules

(R1)	$(\phi[\varphi_1 \supset \varphi_2]_\xi, \xi.n.\iota)$	$\to (\phi[\xi/\neg\varphi_1 \vee \varphi_2], \xi.\iota'.n.\iota)$
	where $\iota' = 1$ if $n = 1$ and else $\iota' = \epsilon$	
	$(\phi[\neg\neg\varphi]_\xi, \xi.1.1.\iota)$	$\to (\phi[\xi/\varphi], \xi.\iota)$
	$(\phi[\neg\exists x\, \varphi]_\xi, \xi)$	$\to (\phi[\xi/\forall x\, \neg\varphi], \xi)$
	$(\phi[\neg\forall x\, \varphi]_\xi, \xi)$	$\to (\phi[\xi/\exists x\, \neg\varphi], \xi)$
	$(\phi[\neg(\varphi_1 \vee \varphi_2)]_\xi, \xi.1.n.\iota)$	$\to (\phi[\xi/\neg\varphi_1 \wedge \neg\varphi_2], \xi.n.1.\iota)$
	$(\phi[\neg(\varphi_1 \wedge \varphi_2)]_\xi, \xi.1.n.\iota)$	$\to (\phi[\xi/\neg\varphi_1 \vee \neg\varphi_2], \xi.n.1.\iota)$
(R2)	$(\phi[\varphi_1 \vee (\varphi_2 \vee P(\overline{t_n}) \equiv \psi)]_\xi, \xi.2.2)$	$\to (\phi[\xi/(\varphi_1 \vee \varphi_2) \vee P(\overline{t_n}) \equiv \psi], \xi.2)$
	$(\phi[\exists x\, (\varphi \vee P(\overline{t_n}) \equiv \psi)]_\xi, \xi.1.2)$	$\to (\phi[\xi/\exists x\, \varphi \vee P(\overline{t_n}) \equiv \psi], \xi.2)$
	if $x \notin \mathit{free}(P(\overline{t_n}))$	
	$(\phi[\forall x\, (\varphi \vee P(\overline{t_n}) \equiv \psi)]_\xi, \xi.1.2)$	$\to (\phi[\xi/\forall x\, \varphi \vee P(\overline{t_n}) \equiv \psi], \xi.2)$
	if $x \notin \mathit{free}(P(\overline{t_n}))$	
	$(\phi[\varphi_1 \vee \forall x\, \varphi_2]_\xi, \xi.2.1.\iota)$	$\to (\phi[\xi/\forall x\, (\varphi_1 \vee \varphi_2)], \xi.1.2.\iota)$
	if $\varphi_2\vert_\iota$ is $P(\overline{t_n}) \equiv \psi$ or $\psi \equiv P(\overline{t_n})$ and $x \in \mathit{free}(P(\overline{t_n}))$	
(R3)	$(\phi[\varphi \vee P(\overline{t_n}) \equiv \psi]_\xi, \xi.2)$	$\to (\phi[\xi/\neg\varphi \supset P(\overline{t_n}) \equiv \psi], \xi.2)$

The algorithm is now composed of successive exhaustive application of each group of transformation rules.

Definition 7 (Extraction algorithm for atom definitions). Given a formula ϕ and a position π in ϕ, the *extraction algorithm* consists of the following stages of successive extraction steps that start from (ϕ, π):
Stage 1: Apply *(R1)* exhaustively.
Stage 2: Apply *(R2)* exhaustively.
Stage 3: Apply *(R3)* if possible.

We now prove that this algorithm is terminating, equivalence preserving, correct and complete with respect to Definition 4.

Theorem 8. The extraction algorithm terminates for any input (ϕ, π) with π a position in the formula ϕ. Furthermore, let (ϕ', π') be the output of the algorithm for input (ϕ, π). Then ϕ and ϕ' are equivalent; and ϕ' is an atom definition for P at π' if and only if ϕ contains an atom definition for P at π.

Proof. (Termination) Within the first stage, termination can already be established if the transformation rules are considered as a rewrite system, interpreting formulae as terms and embedding the system into a recursive path ordering with appropriate precedence. The rules in the second stage either decrease the length of the position, or that length remains unchanged, but the length of a position

of a quantifier for one of the finitely many free variables in the subformula at hand is decreased. Finally, the rule in the third stage is applied no more than once.

(Equivalence preservation.) This follows immediately from Lemma 6.

(Correctness.) One has to show that if the algorithm outputs (ϕ', π') where ϕ' is an atom definition for a predicate P at π', then the input ϕ contains an atom definition for P at π. In fact, the formula ϕ' of course contains an atom definition for P at π'; and one easily shows by inspection of the extraction rules that, if the resulting formula of a transition contains a definition, then the original formula did as well.

(Completeness.) Assume the algorithm starts with a formula ϕ as input which contains an atom definition for P at π. By Lemma 6, the output formula ϕ' contains such a definition at π'. We have to show that ϕ' even adheres to the syntactically stricter form of an atom definition. Let (ϕ_2, π') denote the result of the Stages 1 and 2, i.e. before (possible) single application of the rule of the last stage.

What are the operators that may appear on the path from the root position of the formula ϕ_2 to the position π'? First of all, due to the positive polarity $pol(\phi_2, \pi')$, no equivalence is possible there. Second, because of the exhaustive application of rules in Stage 1, $\phi_2|_\xi$ cannot be an implication for any position ξ above π'. Third, since negations have uniformly been moved downwards, they could only appear immediately above the position π'. Due to the elimination of double negations, their number is 0 or 1, the latter being impossible since $pol(\phi_2, \pi') = 1$. Finally, if there were a conjunction, then because of its negative polarity there should be a negation above it, but there is actually none. So $\phi_2|_\xi$ is a disjunction or a quantification for all $\xi < \pi'$.

Let us now have a look at a quantification $\phi_2|_\xi$ at a position ξ above π'. Because of Stage 2 it must bind a variable in $\phi_2|_{\pi'}$. Since there are no more negations, its polarity is positive, hence it can only be a universal one. In fact, since every free variable in the equivalence $\phi_2|_{\pi'}$ is bound, there is one quantification for each free variable thereof. These quantifications do not appear within disjunctions, i.e. they form some prenex, whereas the underlying disjunction must contain the equivalence $\phi_2|_{\pi'}$ as right top-level disjunct. So ϕ_2 is a formula of the form $\overline{\forall x_k}\,(\varphi' \vee P(\overline{t_n}) \equiv \psi)$. Finally Stage 3 transforms ϕ_2 into $\phi' = \overline{\forall x_k}\,(\varphi \supset P(\overline{t_n}) \equiv \psi)$ where $\varphi = \neg\varphi'$, which indeed is an atom definition for P. □

The notions developed so far can straightforwardly be extended to cover also *conjunctive occurrence* of subformulae, and thereby to handle atom definitions contained in a conjunctive fashion.

Finally, note that the extraction algorithm presented here is in principle also applicable to transform the contexts in which an atom definition is to be applied. One starts with the corresponding notion of a formula containing an atom occurrence and modifies Stage 2 such that it works on occurrences instead of equivalences.

5 Experiments

Several problem domains are naturally formalized using atom definitions, like software verification/analysis or set theory. We concentrate on the latter, because the examples presented below are widely available through the TPTP [4]. The current version of the TPTP at this writing is 2.4.1.

The previous chapter leaves open any aspects of an actual implementation of the theory. We found the following settings to be useful: (i) all found atom definitions are only applied to the conjecture formula, (ii) for the proof of the guard formula, we use SPASS itself restricted to depth-bounded unit resolution [5], (iii) the number of atom definition applications is limited to at most 10 applications,[3] (iv) all atom definitions are kept, even if the defined predicate could be completely eliminated through expansions. These features are available in SPASS Version 2.0 that is available from the SPASS homepage: `http://spass.mpi-sb.mpg.de/`.

The hardware used for the test are old-fashioned PCs (333MHz Pentium II) running SUN OS 5.6, so the results should be easily reproducible on any current hardware. The examples shown in Table 2 are all examples that SPASS could not solve without detecting and expanding atom definitions in the way described before. These examples make up about 20% of all first-order SET examples in the TPTP. The table shows the TPTP problem name together with the time in seconds needed for SPASS to prove the problem. All numbers are upwards rounded to seconds. Summing it up, with the techniques described in this paper, SPASS can solve in a few seconds many problems that without them could not be done at all within any reasonable time limit. For many of these examples, SPASS was the first automatic prover to report a successful proof attempt to the TPTP; so we encourage the reader to test these examples using his/her favorite theorem proving system.

In general, it is pretty clear that changes to any parameter chosen for this experiment have influence on the success or non-success of a particular proof attempt. It is well known from proof attempts in mathematics that there are problems where the expansion of definitions in the conjecture is the key to a successful proof and that there are problems where just the opposite is the case.

Nevertheless, we made some additional runs on the TPTP SET first-order problems that may give a feeling for the influence the parameters have on SPASS's success. If we change the number of atom definition applications from 10 to 5, then we loose 15 out of the above 60 problems. If we change it from 10 to 15, then we loose 2 problems. However, beside the above 60 problems, both parameter settings can prove further problems. For all experiments we made on the TPTP SET domain, the time needed by SPASS to actually extract the atom definition, the context of the atom to be replaced and to prove the guard formula could be neglected.

If we compare SPASS with and without expansion of atom definitions on all SET first-order examples of the TPTP except the above 60 problems, the

[3] We will discuss this "magic number" at the end of the section.

Table 2. SET Experiments

Problem	time	Problem	time	Problem	time	Problem	time	Problem	time
SET010+3	1	SET011+3	1	SET012+4	1	SET013+4	1	SET016+4	1
SET066+1	2	SET069+1	192	SET071+1	1	SET143+3	1	SET143+4	1
SET144+3	1	SET155+4	2	SET156+4	5	SET159+3	1	SET159+4	1
SET169+3	1	SET169+4	1	SET171+3	1	SET171+4	1	SET173+3	1
SET175+3	1	SET351+4	1	SET352+4	1	SET358+4	21	SET593+3	1
SET595+3	1	SET606+3	1	SET607+3	1	SET608+3	1	SET609+3	1
SET610+3	1	SET611+3	1	SET612+3	1	SET613+3	1	SET614+3	1
SET615+3	1	SET624+3	101	SET634+3	1	SET690+4	15	SET692+4	1
SET693+4	1	SET694+4	1	SET695+4	1	SET696+4	1	SET698+4	71
SET699+4	1	SET700+4	1	SET701+4	1	SET702+4	1	SET703+4	1
SET706+4	5	SET720+4	31	SET722+4	18	SET731+4	2	SET734+4	18
SET751+4	3	SET753+4	4	SET754+4	4	SET755+4	3	SET763+4	2

expansion version looses 5 problems compared to the version without expansion of atom definitions. However, it also wins several new problems beside the above 60 problems. We didn't include these problems into the above 60 ones, because they can also be solved with Spass without expansion of atom definitions, but other parameter variations.

6 Summary and Related Work

We have extended the notion of atom definitions in first-order logic by guards and presented conditions under which atom definitions can be expanded within formulae. Furthermore, we gave an effective algorithm to retrieve such definitions from equivalences within complex formulae. We presented a syntactic criterion for a formula to contain a definition and showed that our algorithm is correct and complete with respect to the criterion. To the best of our knowledge, there is no previous work in the context of automated theorem proving that relates to these results on atom definitions with guards on the formula level.

The theory is implemented within the CNF translation module of Spass. Feeding its outcome into Spass, we have performed an experimental evaluation showing that the prover can benefit tremendously in application domains where atom definitions with guards are frequent. Our approach being that successful already, future work will focus on the question of expanding atom definitions with guard dynamically also in the theorem proving process itself.

On the clause level, there exist contributions how definitions can be detected and utilized. Plaisted and Zhu [3] presented an approach how to detect definitions on the clause level and to turn them into replacement rules. These rules are then employed to simulate the expansion of atom definitions on the clause level. Despite from the fact that the detection of guards on the clause level is not

straightforward, applying this work to atom definitions with guards means that the guards are checked after expansion.

More recently, Degtyarev and Voronkov [1] introduced stratified resolution, a sound and complete resolution calculus that can in particular be used to restrict resolution on clauses resulting from atom definitions such that only expansion steps are preformed.

Acknowledgments. We thank Andrei Voronkov for many discussions and a number of valuable comments.

References

1. A. Degtyarev and A. Voronkov. Stratified resolution. In D. McAllester, editor, *17th International Conference on Automated Deduction (CADE-17)*, volume 1831, pages 365–384, Pittsburgh, 2000.
2. Andreas Nonnengart, Georg Rock, and Christoph Weidenbach. On generating small clause normal forms. In Claude Kirchner and Hélène Kirchner, editors, *15th International Conference on Automated Deduction, CADE-15*, volume 1421 of *LNAI*, pages 397–411. Springer, 1998.
3. David A. Plaisted and Yunshan Zhu. Replacement rules with definition detection. In Ricardo Caferra and Gernot Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics*, volume 1761 of *LNAI*, pages 80–94. Springer, 1998.
4. Geoff Sutcliffe, Christian B. Suttner, and Theodor Yemenis. The TPTP problem library. In Alan Bundy, editor, *Twelfth International Conference on Automated Deduction, CADE-12*, volume 814 of *LNAI*, pages 252–266, Nancy, France, June 1994. Springer.
5. Christoph Weidenbach. Combining Superposition, Sorts and Splitting. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, chapter 27, pages 1965–2013. Elsevier Science Publishers B.V., 2001.
6. Larry Wos. *Automated Reasoning, 33 Basic Research Problems*. Prentice Hall, 1988.