A Tableaux-based calculus for Abduction in Expressive Description Logics: Preliminary Results

Tommaso Di Noia (1), Eugenio Di Sciascio (1), Francesco M. Donini (2)

(1) SisInfLab, Politecnico di Bari, Bari, Italy (2) Università della Tuscia , Viterbo, Italy
{t.dinoia,disciascio}@poliba.it,donini@unitus.it

1 Introduction

Abduction [12] is a well-known form of common-sense reasoning that has been widely exploited in Artificial Intelligence applications, including formalization of Diagnostic Reasoning [13] and Scene Interpretation [14]. Abduction has been also proposed as a logical tool for formalizing hypothetical reasoning in E-Commerce [7], Negotiation [15] and Web-service Discovery [10], among others. The computation of Propositional Abduction (PA) has been extensively studied, for nearly all criteria that can be adopted for choosing the preferred hypotheses [6]. Also Abductive Logic Programming is well established [9], and its computation resorts on a modification of SLD-resolution. Yet, when the representation language chosen for the application domain is a Description Logic (DL), the picture is far from being complete. Since concepts in a DL are in fact monadic predicates, the definition of Abduction should be extended from a propositional to a first-order setting. A possible definition has been proposed by Di Noia et al. [7]: given concepts C (prior facts), D (target), and a TBox $\mathcal{T}$ (background knowledge), a Concept Abduction Problem is finding another concept H such that both *(i)* the conjunction of C and H—denoted in DL by $C \sqcap H$—is satisfiable in $\mathcal{T}$, and *(ii)* in all models of $\mathcal{T}$, $C \sqcap H$ is subsumed by D (denoted by $\mathcal{T} \models C \sqcap H \sqsubseteq D$). A tableaux-based calculus for computing Concept Abduction in the DL $\mathcal{ALN}$ was devised in [5]. However, although $\mathcal{ALN}$ is computationally simple, it is a rather inexpressive DL. We propose here a form of Concept Abduction which is computed extending the framework of Concept Matching [2]. In a nutshell, we identify places in the description of C where a hypothesis can be added, and name such places with (all distinct) concept variables $H_0, H_1, H_2, \ldots$ We denote C^h the resulting concept. Then a solution to such an Abduction Problem is a substitution σ of concept variables with concepts, such that $\mathcal{T} \models \sigma(C^h) \sqsubseteq D$. Note that in Concept Matching the concept with variables is D, and a solution is a substitution σ such that $\mathcal{T} \models C \sqsubseteq \sigma(D)$. The remaining of this paper is structured as follows. In the next section, we lay out some definitions and other preliminary notions. Then we propose a calculus based on tableaux for finding substitutions, and in Section 4 we propose some strategies for finding "good" substitutions. Finally, we compare our proposal with existing literature.

2 Structural Abduction in $\mathcal{SH}$

Here, we admit only unfoldable TBoxes and we denote by $\sqsubseteq^*$ the transitive closure of role inclusion over the RBox $\mathcal{R}$. For the sake of conciseness we assume the RBox as part

of the TBox: $\mathcal{R} \subseteq \mathcal{T}$. We assume that concepts are always in Negation Normal Form (NNF), where negation in front of a concept is always "pushed" inside connectives and quantifiers, recursively, till negation is in front of concept names only. In the following, we augment concept expressions by means of *concept variables*. We call *concept term* [3] every concept expression formed by using the syntax above, plus the ability of using a concept variable $h \in \{h_0, h_1, h_2, \ldots\}$, in place of a concept name. To extend the semantics of $\mathcal{SH}$ to concept terms, we let σ be an assignment $\sigma : h \mapsto C \in \mathcal{SH}$ that interprets concept variables as concepts in $\mathcal{SH}$, and we define the semantics of h as $(\sigma(h))^{\mathcal{I}}$. We recall previous definition of Concept Abduction in $\mathcal{ALN}$ [7].

Definition 1 (Concept Abduction). *Let C, D, be two concepts in $\mathcal{ALN}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{ALN}$, where both C and D are satisfiable in $\mathcal{T}$. A* Concept Abduction Problem *(CAP), denoted as $\langle C, D, \mathcal{T}, \mathcal{L} \rangle^C$, is finding a concept $H \in \mathcal{ALN}$ such that $\mathcal{T} \not\models C \sqcap H \equiv \bot$, and $\mathcal{T} \models C \sqcap H \sqsubseteq D$.*

While adequate for Description Logics as $\mathcal{ALN}$, Def. 1 shows its limits when dealing with languages where qualified existential quantification is allowed. This is the case of $\mathcal{ALE}$ and, more generally, of $\mathcal{ALC}$ and $\mathcal{SH}$. We explain the need to generalize Def. 1 with the aid of a simple example.

Example 1. Let W_1, W_2 be two Web services, W_1 having two payment methods: a Credit-card one and a non-https-based one ($W_1 = \exists \texttt{howPay}.\texttt{CC} \sqcap \exists \texttt{howPay}.\neg\texttt{https}$), and W_2 for which it is only known that it has some payment method ($\exists \texttt{howPay}.\top$). Suppose a user has to choose which service best satisfies her demand $D = \exists \texttt{howPay}.(\texttt{CC} \sqcap \texttt{https})$. Only for sake of simplicity, let the TBox be empty. Obviously, both $W_1 \not\sqsubseteq D$ and $W_2 \not\sqsubseteq D$—*i.e.*, neither service completely fulfills D—so the problem arises as to whether there is some logic-based method to compare them. Some researchers [7, 10] propose to automate the choice by solving two (kind of) abduction problems, and compare the results. Namely, they propose to compare some hypotheses H_1 and H_2 that, when added to W_1 and W_2 respectively, would make each one of them be subsumed by D. Then, an offer with a more generic H should be preferred over one with a more specific H. Following this criterion, an offer W for which $H = \top$ is the "best" choice, since in this case already $W \sqsubseteq D$. Following Def. 1, we would have to find two concepts H_1, H_2 such that $W_i \sqcap H_i \sqsubseteq D$ $(i = 1, 2)$ and in this case it can be verified that both problems admit one subsumption-maximal solution, namely $H_1 = H_2 = \exists \texttt{howPay}.(\texttt{CC} \sqcap \texttt{https}) = D$. Observe that $H_{1,2}$ is trivially hypothesizing the whole target conclusion, disregarding the fact that the conjunct $\exists \texttt{howPay}.\texttt{CC}$ in W_1 already "covers" part of D. That is, Def. 1 cannot be used to prefer an offer that already partly covers the demand (W_1) over another (W_2) that does not[1]. △

The explanation computed in Ex.1 according to Def. 1 involves the whole existential restriction even though only a part of it would be necessary—in this case `https`. This is because Concept Abduction (and all definitions of PA) only adds hypotheses as outermost conjunctions. Instead, solutions should take into account the structure of a formula; this amounts to hypothesize pieces of a formula to be added inside the *quantifiers* of C and not just added as outermost conjunctions.

[1] We remark that also the approach in [4] would compute $\diamond(\texttt{CC} \wedge \texttt{https})$ (with $\diamond$ the modality associated with `howPay`) for both abduction problems. Hence, also their proposal cannot be used to highlight the *missing* information inside quantifications.

Definition 2 (Abducible Concept – Hypotheses List). *Let C and D be two $\mathcal{SH}$-concepts in NNF, and let $\mathcal{T}$ be a set of axioms in $\mathcal{SH}$. We define* abducible concept *$C^h \doteq h_0 \sqcap Rew(C)$, where the rewriting $Rew(C)$ defined recursively as*
$Rew(A) = A$; $Rew(\neg A) = \neg A$; $Rew(C_1 \sqcap C_2) = Rew(C_1) \sqcap Rew(C_2)$
$Rew(\exists R.C) = \exists R.(h_{new} \sqcap Rew(C))$; $Rew(\forall R.C) = \forall R.(h_{new} \sqcap Rew(C))$ *where by h_{new} we mean a concept variable not yet appearing in the rewriting*[2]*. We assume that concept variables are numbered progressively, and call* hypotheses list *of C^h the list $\overline{\mathcal{H}} = \langle h_0, h_1, h_2, \ldots \rangle$.*

Observe that since C is in NNF, a case for $Rew(\neg C)$, with C a generic concept, is not present in the definition of $Rew()$. Given an abducible concept C^h, its corresponding hypotheses list $\overline{\mathcal{H}} = \langle h_0, \ldots, h_\ell \rangle$ and a list $\mathcal{C} = \langle C_0, \ldots, C_\ell \rangle$ of $\mathcal{ALECH}_{R^+}$ concepts, we denote with $\sigma[\overline{\mathcal{H}}/\mathcal{C}]$ the substitution $\{h_0 \mapsto C_0, \ldots, h_\ell \mapsto C_\ell\}$ and with $\sigma_i[\overline{\mathcal{H}}/\mathcal{C}]$, with $i = 0, \ldots, \ell$, the substitution of the single variable h_i.

Definition 3 (Structural Abduction). *Let $C, D \in \mathcal{SH}$, be two concepts in NNF, and $\mathcal{T}$ be a set of axioms in $\mathcal{SH}$, where both C and D are satisfiable in $\mathcal{T}$ and $\mathcal{T} \not\models C \sqcap D \sqsubseteq \bot$. Let $\overline{\mathcal{H}} = \langle h_0, \ldots, h_\ell \rangle$ be the hypotheses list of the abducible concept C^h and $\tilde{\mathcal{A}} = \langle \mathcal{A}_0, \ldots, \mathcal{A}_\ell \rangle$ (for Assumptions) be a list of $\mathcal{SH}$ concept sets. A* Structural Abduction Problem *(SAP) for $\mathcal{SH}$, denoted as $\langle C, D, \mathcal{T}, \tilde{\mathcal{A}} \rangle^S$, is finding a list of concepts $\mathcal{H} = \langle H_0, \ldots, H_\ell \rangle$ in $\mathcal{ALECH}_{R^+}$ such that*

$$H_i \in \mathcal{A}_i \text{ for every } i = 0, \ldots, \ell \tag{1}$$
$$\mathcal{T} \not\models \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \sqsubseteq \bot \tag{2}$$
$$\mathcal{T} \models \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \sqsubseteq D \tag{3}$$

We call a SAP General *when $\mathcal{A}_i = \mathcal{SH}$, for every $i = 0, \ldots, \ell$.*

Following [5] we use $\mathcal{P}^S$ as a symbol for a SAP, and we denote with $SOLSAP(\mathcal{P}^S)$ the set of all solutions to a SAP $\mathcal{P}^S$. Observe that we impose the hypotheses in $\overline{\mathcal{H}}$ to be in the DL $\mathcal{ALECH}_{R^+}$, in which disjunction is not allowed, and negation is allowed only in front of concept names. This restriction is analogous to the one in PA—where the set of abduced hypotheses is always intended as a conjunction—and Abductive Logic Programming—where again the abduced atoms are (implicitly) taken conjunctively. As noted by Marquis [11] for First-order Abduction, if disjunction and full negation are allowed in solutions to an Abduction problem, the most general hypothesis is always $\neg C \sqcup D$, which in fact solves nothing. Hereafter, we always refer to *General SAP*s.

Example 2. Consider W_1 and D as in Ex.1. According to Def. 2 and Def. 3 we have

$$\begin{aligned} W_1^h &= h_0 \sqcap \exists \texttt{howPay}.(\texttt{CC} \sqcap h_1) \sqcap \exists \texttt{howPay}.(\neg \texttt{https} \sqcap h_2) \\ \overline{\mathcal{H}} &= \langle h_0, h_1, h_2 \rangle \\ \mathcal{H} &= \langle \top, \texttt{https}, \top \rangle \\ \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) &= \top \sqcap \exists \texttt{howPay}.(\texttt{CC} \sqcap \texttt{https}) \sqcap \exists \texttt{howPay}.(\neg \texttt{https} \sqcap \top) \end{aligned}$$

Observe that the solution previously computed in Ex.1 is still a solution of the SAP, namely, it is the solution $\mathcal{H}' = \langle \exists \texttt{howPay}.\texttt{CC} \sqcap \texttt{https}, \top, \top \rangle$. △

[2] A precise procedure to obtain this result would need a global counter, visible by all recursive calls. We skip this technical detail to simplify presentation.

Solutions can be compared according to two preference criteria, namely, componentwise subsumption, and subsumption in the abducible concept after substitution.

Definition 4 (Preference and Maximality). *Let $\mathcal{P}^S$ be a SAP and $\mathcal{H}' = \langle C'_0, \ldots, C'_\ell \rangle$ and $\mathcal{H}'' = \langle C''_0, \ldots, C''_\ell \rangle$ be two solutions in $SOLSAP(\mathcal{P}^S)$. We say that:*
— $\mathcal{H}'$ is structurally-*preferred over $\mathcal{H}''$, denoted by $\mathcal{H}' \succ_{\sqsubseteq S} \mathcal{H}''$, if for $i = 0, \ldots, \ell$ we have $\mathcal{T} \models C''_i \sqsubseteq C'_i$;*
— $\mathcal{H}'$ is subsumption-*preferred over $\mathcal{H}''$, denoted as $\mathcal{H}' \succ_{\sqsubseteq} \mathcal{H}''$, if $\mathcal{T} \models \sigma[\overline{\mathcal{H}}/\mathcal{H}''](C^h) \sqsubseteq \sigma[\overline{\mathcal{H}}/\mathcal{H}'](C^h)$;*
A solution is Structural-maximal *if no solution is structurally preferred to it, and* Subsumption-maximal *if no solution is subsumption-preferred to it.*

However, it turns out that subsumption-preference includes structural preference, hence we can safely forget the latter.

Proposition 1. *Given a SAP $\mathcal{P}^S = \langle C, D, \mathcal{T}, \tilde{\mathcal{A}} \rangle^S$ and two solutions $\mathcal{H}'$ and $\mathcal{H}''$ in $SOLSAP(\mathcal{P}^S)$, if $\mathcal{H}' \succ_{\sqsubseteq S} \mathcal{H}''$ then $\mathcal{H}' \succ_{\sqsubseteq} \mathcal{H}''$.*

Proof. Recall that (Def. 3) both C and D are in NNF. For this form, substitutions are always monotonic over $\sqsubseteq$, that is, if $C_1 \sqsubseteq C_2$ then for every $i = 0, \ldots, \ell$, $\sigma_i[h_i/C_1](C^h) \sqsubseteq \sigma_i[h_i/C_2](C^h)$. This is because concept variables appear only in conjunctions, and positively—*i.e.*, inside no negation. Iterating over the list of concept variables, the claim follows. □

Note that the requirement that C is in NNF before rewriting in C^h is fundamental for the maximality criteria. Consider $C = \neg\exists R.A$; without NNF, C^h would be $h_0 \sqcap \neg\exists R.(A \sqcap h_1)$, but due to the fact that h_1 is added as a conjunction inside an odd number of negations, by substituting h_1 with anything different from $\top$ we are in fact making C^h more generic, not more specific. So, in the criteria for subsumption-maximal solutions, we should now distinguish between h's occurring inside an even number of negations and h's occurring inside an odd number of negations (for which the criteria should be reverted: a more generic substitution yields a more specific C^h). We preferred to use NNF, so that each concept variable h occurs inside 0 negations in C^h, and use uniform maximality criteria. Recalling Example2, we observe that none of the previous criteria allows us to prefer $\mathcal{H}$ to $\mathcal{H}'$. This is because $\mathcal{H}$ and $\mathcal{H}'$ are incomparable under structural preference, while $\sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \equiv \sigma[\overline{\mathcal{H}}/\mathcal{H}'](C^h)$. Hence a different preference criterion is needed.

Definition 5. *Given an abducible concept C^h, we introduce the following* preorder *among variables in $\overline{\mathcal{H}}$:(i)$h_0 < h_i$ with $i \neq 0$; (ii) $h_i < h_p$ if h_p is within a quantification inside the one of h_i; (iii) $h_i < h_p$ if h_p is within an existential quantification $\exists R.(h_p \sqcap \cdots)$, h_i is within a universal quantification $\forall R.(h_i \sqcap \cdots)$, and both quantifications appear in the same quantification (*i.e.*, they are siblings in the syntactic tree). Now let $\mathcal{H}' = \langle C'_0, \ldots, C'_\ell \rangle$ and $\mathcal{H}'' = \langle C''_0, \ldots, C''_\ell \rangle$ be two solutions in $SOLSAP(\mathcal{P}^S)$. We say that $\mathcal{H}' \leq \mathcal{H}''$ if $C'_i \sqsubseteq C''_i$ for some i, and for all p such that $h_i < h_p$, $C'_p = C''_p$.*

Example 3. Turning back to Example 2, we remember that $\mathcal{H} = \{\top, \texttt{https}, \top\}$ and $\mathcal{H}' = \{\exists \texttt{howPay}.\texttt{CC} \sqcap \texttt{https}, \top, \top\}$. By Definition 5 we see $\mathcal{H} \leq \mathcal{H}'$. Indeed, we have $\texttt{https} \sqsubseteq \top$ ($C_1 \sqsubseteq C'_1$) for $i = 1$ and $\top = \top$ ($C_2 = C'_2$) for $i > 1$. △

3 Calculus

We present a calculus and algorithms to compute a solution to a SAP as defined in Section 2 for the expressive DL $\mathcal{SH}$. We assume the reader be familiar with tableau calculus [16]. Following [5], we will build a prefixed tableau using two labeling functions (to represent true/false prefixed tableaux) **T**() and **F**() instead of a single labelling one **L**(). **T**() and **F**() map an individual n to a set of concepts $\mathbf{T}(n)$ or $\mathbf{F}(n)$ or relate n to another individual m via a role R. More formally, given two individuals n and m in a tableau τ, the formal semantics of **T**() and **F**() is as follows. We say that an interpretation $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ satisfies two tableau labels $\mathbf{T}(n)$ and $\mathbf{F}(n)$ if:
— for every concept $C \in \mathbf{T}(n)$ and every concept $D \in \mathbf{F}(n)$, $n^{\mathcal{I}} \in C^{\mathcal{I}}$ and $n^{\mathcal{I}} \notin D^{\mathcal{I}}$;
— for every role $R \in \mathbf{T}(n, m)$ and for every role $Q \in \mathbf{F}(n, m)$, $(n^{\mathcal{I}}, m^{\mathcal{I}}) \in R^{\mathcal{I}}$ and $(n^{\mathcal{I}}, m^{\mathcal{I}}) \notin Q^{\mathcal{I}}$;
— an interpretation satisfies a branch $\mathcal{B}$ of τ if it satisfies $\mathbf{T}(n)$, $\mathbf{F}(n)$, $\mathbf{T}(n, m)$ and $\mathbf{F}(n, m)$ for every individual n, and for every pair of individuals n, m in $\mathcal{B}$.
Given two individual names n and m, m is called an R-successor of n if, for some R' with $R' \sqsubseteq^* R$, m is a successor of n and either $R' \in \mathbf{T}(n, m)$ or $\neg R' \in \mathbf{F}(n, m)$. Ancestors are definerd as usual. If n is an ancestor of m, we say m *is blocked by* n if either $\mathbf{T}(m) \subseteq \mathbf{T}(n)$ or $\mathbf{F}(m) \subseteq \mathbf{F}(n)$. In order to compute a solution to a SAP, we will build a prefixed tableau τ using the rules in Fig. 1. Since we use two labelling functions, for each classical tableau rule we have we both a **T**) version and its dual **F**) version. The only optimization technique we include here is lazy unfolding [1, 8]. We assume concepts are always simplified in NNF and we use $\overline{C}$ to represent the NNF of a concept C. Given a $\mathcal{SH}$ concept in NNF, whenever we have a role $Q \in \mathbf{F}(n, m)$, it is of the form $\neg R$. Hence $Q \in \mathbf{F}(n, m)$ means, in fact, $(n^{\mathcal{I}}, m^{\mathcal{I}}) \in R^{\mathcal{I}}$ too. Unfolding rules for TBox axioms are presented in Fig. 2. We already discussed the need of having C in NNF. For what concerns D, it is just a matter of not doubling the tableaux rules (if D were not in NNF, we would need, e.g., a rule for $\sqcap$ and another rule for $\neg(... \sqcap ....)$).
In the following we use the definition of homogeneous and heterogeneous clash as proposed in [5]. We recall here the definition of homogeneous and heterogeneous clash as proposed in [5].

Definition 6 (Clash). *A branch $\mathcal{B}$ contains a* homogeneous clash *if it contains one of the following:*
*1. either $\bot \in \boldsymbol{T}(n)$ or $\top \in \boldsymbol{F}(n)$, for some individual n; 2. either $A, \neg A \in \boldsymbol{T}(n)$ (**T**-homogeneous) or $A, \neg A \in \boldsymbol{F}(n)$ (**F**-homogeneous) for some individual n and some concept name A; $\mathcal{B}$ contains a* heterogeneous clash *if it contains one of the following:*
1. $\boldsymbol{T}(n) \cap \boldsymbol{F}(n)$ contains either A or $\neg A$ for some individual n and some concept name A;

We say a branch $\mathcal{B}$ is *complete* iff for each individual name n occurring in $\mathcal{A}$ no new rule application is possible both to $\mathbf{T}(n)$ and $\mathbf{F}(n)$. A complete branch is *open* if it contains no clash, otherwise it is *closed*. A complete tableau is open if it contains at least one open branch, otherwise it is closed. Soundness and completeness of the calculus follow from the version without prefixes [8].

$\sqcap$ **- rules** :

T) if $C \sqcap D \in \mathbf{T}(n)$, then add both C and D to $\mathbf{T}(n)$.

F) if $C \sqcup D \in \mathbf{F}(n)$, then add both C and D to $\mathbf{F}(n)$.

$\sqcup$ **- rules** :

T) if $C \sqcup D \in \mathbf{T}(n)$, then add either C or D to $\mathbf{T}(n)$.

F) if $C \sqcap D \in \mathbf{F}(n)$, then add either C or D to $\mathbf{F}(n)$.

$\exists$ **- rules** :

T) if $\exists R.C \in \mathbf{T}(n)$, n is not blocked, and n has no R-successor m with either $C \in \mathbf{T}(m)$ or $\neg C \in \mathbf{F}(m)$, then pick up a new individual m, add R to $\mathbf{T}(n, m)$, and let $\mathbf{T}(m) := \{C\}$.

F) if $\forall R.C \in \mathbf{F}(n)$, n is not blocked, and n has no R-successor m with either $C \in \mathbf{T}(m)$ or $\neg C \in \mathbf{F}(m)$, then pick up a new individual m, add $\neg R$ to $\mathbf{F}(n, m)$, and let $\mathbf{F}(m) := \{C\}$.

$\forall$ **- rules** :

T) if $\forall R.C \in \mathbf{T}(n)$, n is not blocked, and there exists an individual m such that m is an R-successor of n, then add C to $\mathbf{T}(m)$.

F) if $\exists R.C \in \mathbf{F}(n)$, n is not blocked, and there exists an individual m such that m is an R-successor of n, then add C to $\mathbf{F}(m)$.

$\forall_+$ **- rules** :

T) if $\forall R.C \in \mathbf{T}(n)$, n is not blocked, with $\texttt{Trans}(R) \in \mathcal{R}$ and:
- $R \sqsubseteq^* S$;
- there exists an individual m such that m is an S-successor of n;

then add $\forall R.C$ to $\mathbf{T}(m)$.

F) if $\exists R.C \in \mathbf{F}(n)$, n is not blocked, with $\texttt{Trans}(R) \in R$ and:
- $R \sqsubseteq^* S$;
- there exists an individual m such that m is an S-successor of n;

then add $\exists R.C$ to $\mathbf{F}(m)$.

Fig. 1. Expansion Rules

4 Algorithm and Substitution Strategies

We observe that given a TBox $\mathcal{T}$ and two concept C and D, it results $\mathcal{T} \models C \sqsubseteq D$ iff the tableau τ starting from $C \in \mathbf{T}(1), D \in \mathbf{F}(1)$ is closed. Moreover, we can say that an heterogeneous clash in τ represents in some way an "interaction" between the concepts C and D. An heterogeneous clash can be seen a "clue" that the relation $\mathcal{T} \models C \sqsubseteq D$ might hold. Whenever a complete tableau τ has one or more open branches $\mathcal{B}^j$, if we want the relation $\mathcal{T} \models C \sqsubseteq D$ hold, then we have to force the closure of all open branches $\mathcal{B}^j$ in τ. We observe that if we had a tableau τ' starting with $C \in \mathbf{T}(1)$, and τ' is closed then $\mathcal{T} \models C \sqsubseteq \bot$, *i.e.*, C is unsatisfiable w.r.t. $\mathcal{T}$. Also notice that in this case, since we have only $\mathbf{T}()$ label, all branches will close with a $\mathbf{T}$-homogeneous clash.

Proposition 2. *Let C, D be two concepts in $\mathcal{SH}$, $\mathcal{T}$ be a TBox in $\mathcal{SH}$ with $\mathcal{T} \models C \sqsubseteq \bot$. A complete tableaux τ starting with $C \in \boldsymbol{T}(1)$ and $D \in \boldsymbol{F}(1)$, is such that for each branch $\mathcal{B}$ there is at least one $\boldsymbol{T}$-homogeneous clash.* Proof. *If $\mathcal{T} \models C \sqsubseteq \bot$, and τ is complete, as a direct consequence, it will surely close all its branches with at least one $\boldsymbol{T}$-homogeneous clash.* $\square$

$\sqsubseteq$ **- rules** :

T) if n is an individual such that $A \in \mathbf{T}(n)$ in the branch, and $A \sqsubseteq C \in \mathcal{T}$, then add C to $\mathbf{T}(n)$.

F) if n is an individual such that $\neg A \in \mathbf{F}(n)$ in the branch, and $A \sqsubseteq C \in \mathcal{T}$, then add $\overline{C}$ to $\mathbf{F}(n)$.

$\equiv$ **- rules** :

T) if n is an individual such that $A \in \mathbf{T}(n)$ in the branch, and $A \equiv C \in \mathcal{T}$, then add C to $\mathbf{T}(n)$.

T) if n is an individual such that $\neg A \in \mathbf{T}(n)$ in the branch, and $A \equiv C \in \mathcal{T}$, then add $\overline{C}$ to $\mathbf{T}(n)$.

F) if n is an individual such that $\neg A \in \mathbf{F}(n)$ in the branch, and $A \equiv C \in \mathcal{T}$, then add $\overline{C}$ to $\mathbf{F}(n)$.

F) if n is an individual such that $A \in \mathbf{F}(n)$ in the branch, and $A \equiv C \in \mathcal{T}$, then add $\neg A$ to $\mathbf{F}(n)$.

Fig. 2. Unfolding Rules

Note that in this case, in order to catch the inconsistency of C, we need τ to be complete. Given two concepts C, D and a TBox $\mathcal{T}$ in $\mathcal{SH}$, we call **H-Tableau** a complete tableau built starting from $C^h \in \mathbf{T}(1)$ and $D \in \mathbf{F}(1)$.

Example 4. Suppose to have the following SAP: $C = \exists R.(A_1 \sqcap (A_2 \sqcup \neg A_3))$; $D = \exists R.(A_1 \sqcap A_2) \sqcap \exists R.(A_3 \sqcup \neg A_2)$; $\mathcal{T} = \emptyset$ The tableau computed following rules in Fig. 1 and Fig. 2 is depicted in Fig. 3.

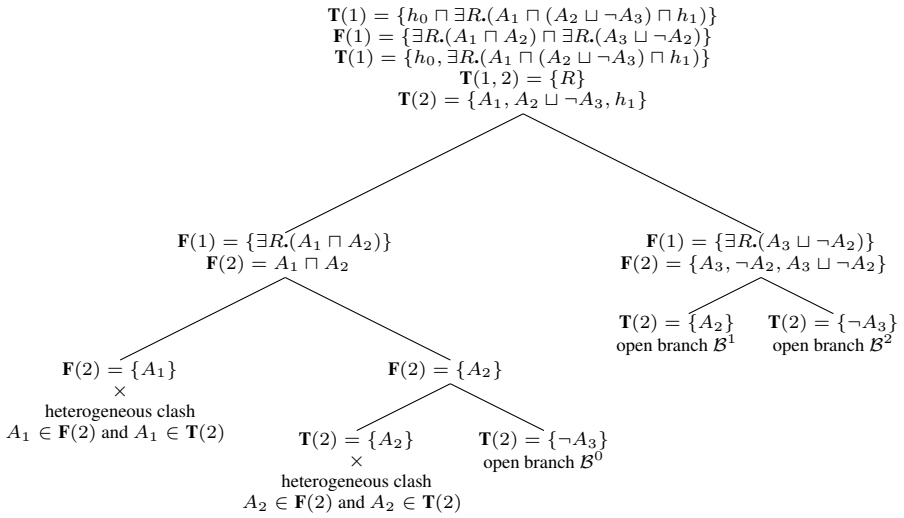

Fig. 3. A H-Tableau computed applying expansion rules in Fig. 1 and Fig. 2.

Looking at Fig. 3 we observe that we could close the tableau by substituting the two concept variables h_0 and h_1 with concepts generating a clash in the open branches $\mathcal{B}^0$, $\mathcal{B}^1$ and $\mathcal{B}^2$. Once the tableau is closed we know that $\mathcal{T} \models \sigma(C^h) \sqsubseteq D$. Actually this relation holds even when trivially $\mathcal{T} \models \sigma(C^h) \sqsubseteq \bot$. Nevertheless, if we want that σ be a solution to a SAP, by condition (2) of Def. 3, we have to avoid that. By Proposition 2, we know that in order to have $\mathcal{T} \not\models \sigma(C^h) \sqsubseteq \bot$, it suffices to have at least one branch closed by a heterogeneous clash. Hence, in order to compute σ we will look for

instantiation of variable in the hypotheses list $\overline{\mathcal{H}}$ of C^h such that, given a complete open tableau τ, there is at least one open branch $\mathcal{B} \in \tau$ such that $\sigma(\mathcal{B})$ closes without any homogeneous clash.

Proposition 3. *Let $\mathcal{T}$ be a TBox, C^h be an abducible concept, $\overline{\mathcal{H}} = \langle h_0, \ldots, h_l \rangle$ be its corresponding hypotheses list and τ be an open complete tableau starting from $C^h \in \boldsymbol{T}(1), D \in \boldsymbol{F}(1)$. For each open branch $\mathcal{B}^j \in \tau$, with $j \in [0, \ldots, k]$, if $\sigma^j[\overline{\mathcal{H}}/\mathcal{H}^j]$ is a substitution such that $\sigma^j[\overline{\mathcal{H}}/\mathcal{H}^j](\mathcal{B}^j)$ is closed then*

$$\sigma[\overline{\mathcal{H}}/\mathcal{H}] = \{h_0 \mapsto \sigma_0^0[\overline{\mathcal{H}}/\mathcal{H}^0] \sqcap \ldots \sqcap \sigma_0^k[\overline{\mathcal{H}}/\mathcal{H}^k], \ldots, h_l \mapsto \sigma_l^0[\overline{\mathcal{H}}/\mathcal{H}^0] \sqcap \ldots \sqcap \sigma_l^k[\overline{\mathcal{H}}/\mathcal{H}^k]\}$$

*is such that $\mathcal{T} \models \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \sqsubseteq D$. We call $\sigma^j[\overline{\mathcal{H}}/\mathcal{H}^j]$ a **branch substitution**. Proof. If $\sigma_i^{\hat{j}}[\overline{\mathcal{H}}/\mathcal{H}^{\hat{j}}]$ closes $\mathcal{B}^{\hat{j}}$ then, by $\sqcap$-**rule T**) in Fig. 1, $\mathcal{B}^{\hat{j}}$ is closed also by $\sigma_i^{\hat{j}}[\overline{\mathcal{H}}/\mathcal{H}^{\hat{j}}] \sqcap C$, being C a generic concept. If $C = \sqcap_{j \neq \hat{j}} \sigma_i^j[\overline{\mathcal{H}}/\mathcal{H}^j]$ the proposition holds.* □

For the sake of clarity, when no confusion arises, from now on we will write σ to denote $\sigma[\overline{\mathcal{H}}/\mathcal{H}]$, σ_i to denote $\sigma_i[\overline{\mathcal{H}}/\mathcal{H}]$, σ^j to denote $\sigma^j[\overline{\mathcal{H}}/\mathcal{H}^j]$ and σ_i^j to denote $\sigma_i^j[\overline{\mathcal{H}}/\mathcal{H}^j]$. Algorithm 1 shows how to compute a solution to SAPs using prefixed tableaux.

Some Desiderata for $\sigma[\overline{\mathcal{H}}/\mathcal{H}]$: In Algorithm 1 we propose a greedy procedure to

Algorithm 1: An Algorithm to compute a solution to a SAP

Algorithm:*abduce*

Input: $\mathcal{SH}$ concepts C, D, acyclic TBox $\mathcal{T}$
Output: substitution σ

```
1  begin
2      σ := {H_0 ↦ ⊤, ..., H_n ↦ ⊤};
3      compute a H-Tableau;
4      if τ is not closed then
5          repeat
6              Θ := ∅;
7              foreach open branch B^j ∈ τ do
8                  find a branch substitution σ^j such that B^j is closed with a heterogeneous
                   clash;
9                  Θ := Θ ∪ {σ^j};
10             end
11             foreach σ^j ∈ Θ do
12                 σ_i := σ_i ⊓ σ_i^j;
13             end
14         until T ⊭ σ(C^h) ⊑ ⊥ ;
15     end
16     return σ;
17 end
```

compute a solution to a General SAP $\langle C, D, \mathcal{T}, \tilde{\mathcal{A}} \rangle^S$. There a practical problem still

remains: the computation of the branch substitution σ^j in line 8. There are different strategies to compute σ^j depending on the characteristics we desire to be shown by σ. In this section we identify and discuss some properties of σ that can be easily computed (implemented) given a H-Tableau, lead by information minimal criteria. In other words, we would like to have a practical (from an implementation point of view) procedure able to compute a solution to a SAP that shows some information minimal properties. The first practical issue we face is: given a H-Tableau, in case we want to solve a General SAP, how could we select concepts in $\mathcal{SH}$ in order to close open branches? We could perform a syntactic search of such concepts looking at the H-Tableau itself:
(C0) For each open branch $\mathcal{B}^j$ in a H-Tableau we select all the sets labeled with $\mathbf{F}(n)$. If there is a concept variable $h_i \in \mathbf{T}(n)$, we could just pick up a concept $C \in \mathbf{F}(n)$ and perform the substitution $\sigma^j_i = \{h_i \mapsto C\}$. It is easy to see that $\sigma^j_i(\mathcal{B}^j)$ closes with at least one heterogeneous clash. **(C1)** For each branch substitution, in order to close a branch it suffices to have at least one variable whose value is different from $\top$[3].
Note that, depending on the substitution we choose, we have different solutions for the same SAP. Suppose we select the substitution $\underline{\sigma}^1 = \{h_0 \mapsto \top, h_1 \mapsto A_3\}$ for $\mathcal{B}^1$ in Fig. 3. It is easy to see that $\underline{\sigma}^1$ closes both $\mathcal{B}^1$ with a heterogeneous clash and $\mathcal{B}^2$ with a homogeneous clash. Hence, in this case we do not need to compute a substitution σ^2 to close $\mathcal{B}^2$. Since we are looking for information minimal hypotheses, in case we are willing to compute, for each variable, *conjunction minimal* substitutions:
(C2) we should avoid variable substitution in conjunctive form. In fact, if a conjunctive substitution $h_i \mapsto D_1 \sqcap D_2$ closes a branch $\mathcal{B}$ then by $\sqcap$-**rule T**) in Fig. 1 either $h_i \mapsto D_1$ or $h_i \mapsto D_2$ closes $\mathcal{B}$; **(C3)** when computing a branch substitution for an open branch, we have to take into account also the substitutions of the other open branches. With respect to Example 4 we see that there are three different substitutions for the same variable h_0. By Proposition 3 we know that the conjunction $\forall R.A_2 \sqcap \forall R.(A_3 \sqcup \neg A_2) \sqcap \exists R.(A_3 \sqcup \neg A_2)$ is a substitution for h_0 closing $\mathcal{B}^0$, $\mathcal{B}^1$ and $\mathcal{B}^2$. Hence, if a variable substitution closes more than one open branch, *i.e.*, the same variable substitution appears in more than one branch substitution, it should be preferred over the others. This is the case, for instance, of $h_0 \mapsto \forall R.(A_3 \sqcup \neg A_2)$ for σ^1 and σ^2;
Going back to Condition (C0) and Condition (C1), given a H-Tableau, in case there is more than one individual n such that $h_i \in \mathbf{T}(n)$, how to choose the individual? What is the best candidate? Moreover, once we choose and individual n, which concept in $\mathbf{F}(n)$ should we pick-up? It depends on the solution we want to compute.
(C4) With respect to Example 4, if we consider either σ^0 in row 3 or σ^0 in row 4 and σ^1 in row 11 we obtain[4]:

$$\sigma[\overline{\mathcal{H}}/\mathcal{H}''] = \{h_0 \mapsto \top, h_1 \mapsto A_2 \sqcap A_3\} \tag{4}$$

$$\sigma[\overline{\mathcal{H}}/\mathcal{H}'] = \{h_0 \mapsto \exists R.(A_1 \sqcap A_2), h_1 \mapsto A_3\} \tag{5}$$

Note that both $\mathcal{H}'$ and $\mathcal{H}''$ are computed satisfying all the above conjunction minimal conditions. Nevertheless, we see that solutions (4) is more "fine-grained" than solu-

[3] Branch substitutions with all the variables in $\mathcal{H}$ but one equal to $\top$ tend to generate structural maximal solutions.

[4] In this case it suffices to have $h_1 \mapsto A_3$ in σ^1 to close both $\mathcal{B}^1$ and $\mathcal{B}^2$.

tion (5). Also notice that $\sigma[\overline{\mathcal{H}}/\mathcal{H}'](C^h) \sqsubseteq \sigma[\overline{\mathcal{H}}/\mathcal{H}''](C^h)$. In other words, solutions (4) is subsumption-maximal w.r.t. solution (5). With respect to Definition 5 it results $\mathcal{H}' \leq \mathcal{H}''$. Hence, when closing an open branch $\mathcal{B}$ to compute a subsumption maximal solution we tend to select variable $h_i \in \mathbf{T}(n)$ such that there is no $h_p \in \mathbf{T}(m)$ with $h_i < h_p$. In Algorithm 2 we propose a procedure to compute a branch substitution σ^j taking into account (C0), (C1), (C2), (C3) and (C4). Moreover, from Proposition 2 we know that if there is at least one branch $\mathcal{B}^j$ in a H-Tableau such that $\mathcal{B}^j$ does not contain any **T**-homogeneous clash, then $\mathcal{T} \not\models \sigma(C^h) \sqsubseteq \bot$. In order to catch this situation as soon as possible, while computing σ^j we should avoid, if possible, to close $\mathcal{B}^j$ generating also a **T**-homogeneous clash.[5]

— In line 1 of Algorithm 2 we take into account Condition (C3). Indeed, given $\mathcal{B}^j$ we try to reuse a substitutions already computed for previous branches $\mathcal{B}^{j-k}$.

— Line 4 formalizes conditions (C0) and (C4). We select an individual n such that both $\mathbf{T}(n)$ contains h_i and there is no other h-variable h_p such that it is "bigger" than h_i.

— Since we selected n such that $\mathbf{F}(n) \neq \emptyset$, we can choose a concept in $\mathbf{F}(n)$ to close $\mathcal{B}^j$. Furthermore, while closing $\mathcal{B}^j$ we try to avoid heterogeneous clashes in order to discard inconsistent substitutions (see Proposition 2).

It is noteworthy that the algorithms we presented in this section compute a sub-optimal

Algorithm 2: Computation of a branch substitution for information minimal solutions.

1 **if** $h_i \in \mathbf{T}(n)$ *and there exists a substitution* σ_i^{j-k}*, with* $k = 1, \ldots, j$ *such that* $\sigma_i^j := \sigma_i^{j-k}$ *closes* $\mathcal{B}^j$ **then**
2 $\sigma_i^j := \sigma_i^{j-k}$;
3 **else**
4 **select** an individual n such that $\mathbf{T}(n) \neq \emptyset$,$\mathbf{F}(n) \neq \emptyset$ and there exists no other individual m such that both $\mathbf{T}(m) \neq \emptyset$, $\mathbf{F}(m) \neq \emptyset$ and $h_i < h_p$ with $h_i \in \mathbf{T}(n)$ and $h_p \in \mathbf{T}(m)$;
5 **choose** a concept $E \in \mathbf{F}(n)$ such that E does no contain a conjunction and, *if possible*, $\mathcal{B}^j \cup \{E \in \mathbf{T}(n)\}$ does not close with a **T**-homogeneous clash;
6 $\sigma_i^j := E$;
7 **end**
8 **foreach** $h_k \in \overline{\mathcal{H}}$ *with* $k \neq i$ **do**
9 $\sigma_k^j := \top$;
10 **end**

solution due to their greedy nature. In fact, we have no guarantee that Algorithm 2 finds a substitution such that $\sigma[\overline{\mathcal{H}}/\mathcal{H}](\tau)$ have at least one branch closed by no **T**-homogeneous clash. However, the algorithms we illustrated in this section are very useful to understand the issues related to the computation of a solution to a SAP.

Dealing with RBoxes: It is easy to see that Algorithm 2 works nicely when $\mathcal{R} = \emptyset$. Let us take a look at what happens when we have to deal with a non-empty RBox.

[5] We minimize **T**-homogeneous clashes in order to avoid trivial solutions.

Example 5. Suppose we have to solve the following SAP. $C = \forall R.(A_1 \sqcap \exists S.A_2)$; $D = \forall T.\exists S.A_3$; $\mathcal{T} = \emptyset$; $\mathcal{R} = \{\texttt{Trans}(R), S \sqsubseteq R, T \sqsubseteq R\}$ Part of the tableau we compute following rules in Fig. 1 is represented in Fig. 4.

$$\begin{array}{c}
\mathbf{T}(1) = \{h_0 \sqcap \forall R.(h_1 \sqcap A_1 \sqcap \exists S.(h_2 \sqcap A_2))\} \\
\mathbf{F}(1) = \{\forall T.\exists S.A_3\} \\
\mathbf{T}(1) = \{h_0, \forall R.(h_1 \sqcap A_1 \sqcap \exists S.(h_2 \sqcap A_2))\} \\
| \\
\mathbf{F}(1,2) = \{\neg T\} \\
\mathbf{F}(2) = \{\exists S.A_3\} \\
\mathbf{T}(2) = \{h_1, A_1, \exists S.(h_2 \sqcap A_2), \forall R.(h_1 \sqcap A_1 \sqcap \exists S.(h_2 \sqcap A_2))\} \\
| \\
\mathbf{T}(2,3) = \{S\} \\
\mathbf{T}(3) = h_2, A_2, h_1, A_1, \exists S.(h_2 \sqcap A_2), \forall R.(h_1 \sqcap A_1 \sqcap \exists S.(h_2 \sqcap A_2)) \\
\mathbf{F}(3) = A_3 \\
\ldots \quad \ldots
\end{array}$$

Fig. 4. Part of the tableau computed for Example 5

Looking at Example 5 we see that, due to $\mathcal{R}$, both h_1 and h_2 appears in $\mathbf{T}(3)$. Hence, following Algorithm 2 we can close the branch with one of the following two substitutions: $\sigma[\overline{\mathcal{H}}/\mathcal{H}'] = \{h_0 \mapsto \top, h_1 \mapsto \top, h_2 \mapsto A_3\}$; $\sigma[\overline{\mathcal{H}}/\mathcal{H}''] = \{h_0 \mapsto \top, h_1 \mapsto A_3, h_2 \mapsto \top\}$
Due to $\texttt{Trans}(R)$, we have $\sigma[\overline{\mathcal{H}}/\mathcal{H}''](C^h) \sqsubseteq \sigma[\overline{\mathcal{H}}/\mathcal{H}'](C^h)$. Similarly to what we do in (C4), if we are looking for subsumption-maximal solutions we will prefer $\mathcal{H}'$ over $\mathcal{H}''3$. On the other hand we see that $\mathcal{H}' \leq \mathcal{H}''$. Hence, (C4) holds also in presence of a non-empty RBox.

5 Remarks and Conclusion

We claim that our proposal is a proper extension of PA. In fact, given a theory T, some observations O (both propositional formulas) and a set of possible assumptions $\mathcal{A} = \{a_1, \ldots, a_n\}$ (atoms), PA is the problem of finding a subset $S \subseteq \mathcal{A}$ such that $T \wedge S$ is consistent, and $\mathcal{T} \wedge S \models O$. Then, PA can be translated into SAP by *(i)* letting $C = \top$ (hence $C^h = \top \sqcap H_0 \equiv H_0$), *(ii)* expressing T and O as $\mathcal{ALC}$ concepts—just replace $\wedge, \vee$ with $\sqcap, \sqcup$—and *(iii)* solving the SAP $\mathcal{T} \models H_0 \sqsubseteq O$, where $\mathcal{T} = \{\top \sqsubseteq T\}$. Since in propositional Logic the Deductive Theorem holds, one could also let $C = T$, $\mathcal{T} = \emptyset$, and solve the SAP $\models T \sqcap H_0 \sqsubseteq O$. In both cases only substitutions involving atoms in $\mathcal{A}$ should be considered. Hence, SAP properly extends PA. Given that $\mathcal{ALC}$ is a syntactic variant of the Modal Logic $\mathbf{K_n}$, and that (the Modal Logic version of) transitive roles is present in the Modal Logic $\mathbf{S4}$, the work most similar to ours is [4]. Compared to our work, Cialdea & Pirri did not have our known facts C. Moreover, they looked for formulas α such that $\mathcal{T} \cup \{\alpha\} \models O$, that in our setting could be rephrased as CA of Section 2, or alternatively, as a SAP having on H_0 as variable to be substituted. Finally, they just looked for subsumption-maximal abductions on H_0, while our approach allows us to find several abduction solutions depending on the preference criterion for the various H_i's. On the other hand, every solution we find can be turned into a solution in their framework: in fact, given a solution $\sigma[\overline{\mathcal{H}}/\mathcal{H}]$, since $\mathcal{T} \models \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h) \sqsubseteq D$, one can always let $\alpha = \sigma[\overline{\mathcal{H}}/\mathcal{H}](C^h)$ as a solution.

References

1. F. Baader, B. Hollunder, B. Nebel, H. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems or "making KRIS get a move on". In *proc. of KR'92*, 1992.
2. F. Baader, R. Küsters, A. Borgida, and D. L. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3), 1999.
3. F. Baader and P. Narendran. Unification of concept terms in description logics. *Journal of Symbolic Computation*, 31(3), 2001.
4. M. Cialdea Mayer and F. Pirri. Modal propositional abduction. *Journal of the IGPL*, 3(6), 1995.
5. S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. A Uniform Tableaux-Based Method for Concept Abduction and Contraction in Description Logics. In *proc. of ECAI'04*, 2004.
6. N. Creignou and B. Zanuttini. A complete classification of the complexity of propositional abduction. *SIAM J. Comput.*, 36(1), 2006.
7. T. Di Noia, E. Di Sciascio, and F. M. Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research*, 29:269–307, 2007.
8. I. Horrocks. Using an expressive description logic: FaCT or fiction? In *proc. of KR '98*, 1998.
9. A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6), 1992.
10. F. Lécué, A. Delteil, and A. Léger. Applying abduction in semantic web service composition. In *proc. of ICWS'07*.
11. P. Marquis. Extending abduction from propositional to first-order logic. In *proc. of FAIR'91*, 1991.
12. C. Peirce. Abduction and induction. In *Philosophical Writings of Peirce*. J. Buchler, 1955.
13. H. E. Pople. On the mechanization of abductive logic. In *proc of IJCAI'73*, 1973.
14. R. Reiter and A. K. Mackworth. A logical framework for depiction and image interpretation. *Artificial Intelligence*, 41, 1989.
15. C. Sakama and K. Inoue. Negotiation by abduction and relaxation. In *proc. of AAMAS'07*, 2007.
16. R. M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.