Appendix A FO(·) to FO transformations

In this section, we provide details on the strong FO(·)-to-FO transformation (see Definition 2) for aggregates and inductive definitions.

Aggregates. For aggregates, we build on ideas in [Pelov 2004]. We only consider decomposable aggregate functions, i.e., aggregates for which $agg(S) = agg(\{agg(S'), agg(S\setminus S')\})$, for sets of domain elements $S$ and $S' \subseteq S$. Our transformation of $agg(\{x \in T : \varphi : t\})$ is based on the decomposability of the considered aggregates and the fact that there is a total order on all domain elements, with a function $MIN$ (minimum) and a (partial) function $PRED$ (predecessor) and that this order can be extended to a total order over tuples of domain elements. Assuming $neutral_{agg}$ is the neutral element of $agg$, we define an accumulator function $acc(T) : \mathbb{N}$ as:

\[
\begin{align*}
acc(MIN(T)) &= t[x/MIN(T)] \quad \leftarrow \varphi[x/MIN(T)] \\
acc(MIN(T)) &= neutral_{agg} \quad \leftarrow \neg\varphi[x/MIN(T)] \\
\forall \pi \in T : acc(\pi) &= agg(acc(PRED(\pi)), t) \quad \leftarrow \varphi \land \neg(\pi = MIN(T)) \\
\forall \pi \in T : acc(\pi) &= acc(PRED(\pi)) \quad \leftarrow \neg\varphi \land \neg(\pi = MIN(T))
\end{align*}
\]

This definition is equivalent to the FO formulas

\[
\begin{align*}
acc(MIN(T)) &= v \equiv (v = t[\pi/MIN(T)] \land \varphi[\pi/MIN(T)]) \lor \\
&\quad (v = neutral_{agg} \land \neg\varphi[\pi/MIN(T)]) \text{ and} \\
acc(\pi) &= v \equiv \neg(\pi = MIN(T)) \land \\
&\quad (v = agg(acc(PRED(\pi)), t) \land \varphi) \lor (v = acc(PRED(\pi)) \land \neg\varphi)
\end{align*}
\]

The aggregate term itself is then replaced by $acc(MAX(T))$.

Inductive definitions. The completion of a definition has models that are not well-founded when the definition contains positive loops. To eliminate such models, we use the idea of level mappings that is used in [Janhunen et al. 2009] for converting ground rule sets to propositional logic and was elaborated for well-founded semantics in [Pelov and Ternovska 2005]. The idea is to introduce a function symbol $l_P(T) : \mathbb{N}$ for every defined
the interpretation of predicate \( P(T) \) (we only work out the predicate case in detail). This
function is axiomatised to be 0 if \( P(T) \) is false, and otherwise it states that \( l_P(T) \) is
strictly larger than the \( l_Q(...) \) of atoms that were positively used to derive the truth of
\( P(T) \). Interpretations that satisfy these constraints, together with the completion, do not
contain positive or mixed loops.

Without loss of generality, we can assume that each predicate (or function) is defined
by a single rule and that it suffices to consider the following cases. For simplicity, assume
\( l_P(T) : N \) also exists for all non-defined symbols \( P(T) \), mapping to 0 for each tuple in \( T \).

- \( \forall x : \neg P(x) \Rightarrow l_P(x) = 0 \) (we assume all well-founded models are total).
- \( \forall x : P(x) \Leftarrow \neg Q(x) \). The constraint is \( \forall x : P(x) \Rightarrow l_P(x) \geq 0 \).
- \( \forall x,y : P(x) \Leftarrow \forall y : Q(x, y) \). The constraint is \( \forall x,y : P(x) \Rightarrow l_P(x) > l_Q(x, y) \).
- \( \forall x : P(x) \Leftarrow \exists y : Q(x, y) \). The constraint is \( \forall x : P(x) \Rightarrow \exists y : l_P(x) > l_Q(x, y) \).
- \( \forall x : P(x) \Leftarrow Q_1(x) \land \ldots \land Q_n(x) \). The constraint is \( \forall x : P(x) \Rightarrow (Q_1(x) \land l_P(x) > l_Q_1(x) \lor \ldots \lor Q_n(x) \land l_P(x) > l_Q_n(x)) \).
- \( \forall x : P(x) \Leftarrow Q_1(x) \land \ldots \land Q_n(x) \). The constraint is \( \forall x : P(x) \Rightarrow l_P(x) > Q_1(x) \land \ldots \land l_P(x) > l_Q_n(x) \).

The advantage of the strong FO(\( ^\cdot \)) to FO transformation is that equivalence is pre-
served, and hence also all functional dependencies. However, the size of the theory in-
creases, and the theorem prover has to reason on induction over the order of the natural
numbers. The theorem prover used in our current prototype cannot.

Appendix B Proof of Proposition 4

First, we introduce some notations. With \( I \) an interpretation, we denote with \( P^I (f^I) \)
the interpretation of predicate \( P \) (function \( f \)). Slightly abusing notation, with \( d \) a domain
element, we say that \( P(d) \in I \) iff the tuple \( d \in P^I \), and similar for \( f(d) = d \in I \). Given a
term \( t \) with free variable \( x \) and a variable assignment \( \{ x/\bar{d} \} \), we denote with \( t(\{ x/\bar{d} \}^I) \) the
interpretation of \( t \) in \( I \) under that assignment (a domain element). Similar for a formula
\( \varphi \), \( \varphi(\{ x/\bar{d} \}^I) \) is the interpretation of \( \varphi \) under that assignment (a truth value).

Our rewriting performs a number of operations to flatten/unflatten terms. We start
with a lemma handling these operations.

Lemma 1
Flattening/unflattening operations on a theory \( T \) preserve its models.

Proof
We distinguish the following operations:

- Let \( T' \) be derived from \( T \) by the replacement of a rule \( \forall \exists : P(T) \Leftarrow \varphi \) by \( \forall \exists : P(t_1, \ldots, t_{j-1}, y, t_{j+1}, \ldots, t_n) \Leftarrow y = t_j \land \varphi \). Let \( M \) be a model of \( T \); we show that it is also a model of \( T' \). It suffices to show that every \( P(d) \), element of \( M \) supported by the original rule in \( T \), is supported by the new rule in \( T' \). That \( P(d) \) is sup-
ported means there is a variable assignment \( \{ x/\bar{d}_x \} \) such that \( P(\{ x/\bar{d}_x \}^M) = P(d) \)
and that \( \varphi(\{ x/\bar{d}_x \}^M) = \top \). But then \( y = t_j \land \varphi \) is true in \( M \) under the vari-
able assignment \( \{ x/\bar{d}_x, y/t_j(\{ x/\bar{d}_x \}^M) \}^M \). We can conclude from this observation that
\( \{ t_1, \ldots, t_{j-1}, y, t_{j+1}, \ldots, t_n \} \{ x/\bar{d}_x, y/t_j(\{ x/\bar{d}_x \}^M) \}^M = \bar{t}(\{ x/\bar{d}_x \}^M = \bar{d} \) and that
$P(\overline{d})$ is supported by the new rule of $\mathcal{T}'$. Hence, $\mathcal{M}$ is also a model of $\mathcal{T}'$. Similarly, we can show that every model of $\mathcal{T}'$ is a model of $\mathcal{T}$; this completes the proof for this case.

- Let $\mathcal{T}'$ be derived from $\mathcal{T}$ by the replacement of an atom $a[t]$ by $\exists y : a[t/y] \land y = t$. Let $I$ be an interpretation and let $\{\overline{x}/\overline{d}\}$ be a variable assignment for the free variables $\overline{x}$ of $a[t]$. If $a[t]\{\overline{x}/\overline{d}\}$ is true then $a[t/y] \land y = t$ is true in $I$ for the variable assignment $\{\overline{x}/\overline{d}, y/t\{\overline{x}/\overline{d}\}\}$ and hence also $\exists y : a[t/y] \land y = t$ is true in $I$ for the variable assignment $\{\overline{x}/\overline{d}\}$. Similarly, one can show that $a[t]$ is false under variable assignment $\{\overline{x}/\overline{d}\}$ in $I$ iff $\exists y : a[t/y] \land y = t$ is. Hence models are preserved by this rewriting.

- Let $\mathcal{T}'$ be derived from $\mathcal{T}$ by the replacement of $agg(\{\overline{x} : \varphi : t\})$ by $agg(\{\overline{x}, y : \varphi \land y = t : y\})$. Let $\{\overline{x}/\overline{d}\}$ be a variable assignment for which $\varphi$ is true in $I$. In this case $t\\{\overline{x}/\overline{d}\}'$ is added to the set to be aggregated. It follows that $\varphi \land y = t$ is true for exactly one variable assignment extending $\{\overline{x}/\overline{d}\}$, namely $\{\overline{x}/\overline{d}, y/t\{\overline{x}/\overline{d}\}\}$. Hence $y(\overline{x}/\overline{d}, y/t\{\overline{x}/\overline{d}\})'$ is added to the set to be aggregated. But $y(\overline{x}/\overline{d}, y/t\{\overline{x}/\overline{d}\})' = t(\overline{x}/\overline{d})'$ hence the same value is added. Also, for all variable assignments $\{\overline{x}/\overline{d}\}$ for which $\varphi$ is false, $\varphi \land y = t$ is false for all variable assignments extending $\{\overline{x}/\overline{d}\}$ with an assignment for $y$. Hence the rewriting preserves the value of the aggregate expression and the models of the theory.

Now, we can turn our attention to the the replacement of predicate and function symbols by new ones in $\text{dep-reduce}$. Given the previous lemma, we can assume without loss of generality that the arguments of rule heads are distinct variables.

**Proof of Proposition 4**

We distinguish several cases.

- The functional dependency $d \in \{(P(\overline{T}), S, j) \mid \#(S) < n - 1 \land \mathcal{T}'\}$ is derived by replacing everywhere $P(\overline{t})$ by $P_r(\overline{t}_{(j)\mathcal{C}})$ and adding the function $f_d(\overline{t}_{\mathcal{S}}) : T_j$. Let $\mathcal{M}$ be a total interpretation of $\mathcal{T}$. Now construct a total interpretation $\mathcal{M}'$ of $\mathcal{T}'$ by copying $\mathcal{M}$ for all symbols except $P$ and by adding the atoms $P_{r}(\overline{d})_{(j)\mathcal{C}}$ and $f_d(\overline{t}_{\mathcal{S}}) = \overline{d}_{(j)}$ for every atom $P(\overline{d})$ in $\mathcal{M}$. Note that $f_d$ defined in this way is a function because of the functional dependency in $\mathcal{T}$. The expressions $P(\overline{t})$ and $P_r(\overline{t}_{(j)\mathcal{C}})$ have the same free variables, say $\overline{x}$. Moreover, given the relationship between $\mathcal{M}$ and $\mathcal{M}'$, it holds that $P(\overline{t})\{\overline{x}/\overline{d}_{\mathcal{C}}\} \land t_j = f_d(\overline{t}_{\mathcal{S}})$ for every variable assignment $\{\overline{x}/\overline{d}_{\mathcal{C}}\}$ hence $\mathcal{M}$ is a model of $\mathcal{T}$ iff $\mathcal{M}'$ is a model of $\mathcal{T}'$. Now, let $\mathcal{M}$ be a model of $\mathcal{T}$ and $\mathcal{M}'$ the corresponding model of $\mathcal{T}'$. Adding the rule $\{\forall \overline{x} \in T : P(\overline{x}) \leftarrow P_r(\overline{x}_{[1,n-1]\{j\}}) \land f_d(\overline{x}_{\mathcal{S}}) = x_j\}$ to $\mathcal{T}'$ as prescribed by Definition 4, $\mathcal{M}'$ is extended with every atom $P(\overline{d})$ that is true in $\mathcal{M}$ and both theories are strongly $\text{voc}(\mathcal{T})$-equivalent.

When $P$ is a defined symbol, every rule $P(\overline{x}) \leftarrow \varphi$ is replaced by the rules $f_d(\overline{x}_{\mathcal{S}}) = x_j \leftarrow \varphi$ and $P_r(\overline{x}_{(j)\mathcal{C}}) \leftarrow \varphi$ (Definition 5). Given the correspondence between $\mathcal{M}$

1 Nothing is added when $t$ contains a function that is not defined for the variable assignment; for that variable assignment, the atom $y = t$ is false.
and $M'$, clearly, an atom $P(\overline{a})$ is supported under $M$ by the rule in $T$ iff $P_r(\overline{a}_{\{j\}c})$ and $f_d(\overline{a}|_S) = d_j$ are supported under $M'$ by the two rules in $T'$. Hence both theories are also strongly $voc(T)$-equivalent in this case.

- The case of a functional dependency $d \langle P(T), S, j \rangle$ in $T$ with $\#(S) = n - 1$ is very similar and the proof is omitted.
- The functional dependency $d \langle f(\overline{t}) : T_n, S, j \rangle$ holds in $T$ with $\#(S) < n - 1$, $j \neq n$, $n \notin S$, and $T'$ is derived by adding the functions $f_d(T|_S) : T_j$ and $f_r(T|_{\{j\}c}) : T_n$ and replacing everywhere atoms $a[f(\overline{t})]$ by $a[f(\overline{t})/f_r(\overline{t}|_{\{j\}c})] \land t_j = f_d(\overline{t}|_S)$. Also here, given a total interpretation $M$ of $T$, we can construct a total interpretation $M'$ of $T'$ by copying the interpretations of all symbols but $f$ and extending it with $f_d(\overline{a}|_S) = d_j$ and $f_r(\overline{a}|_{\{j\}c}) = d_n$ for every atom $f(\overline{a}) = d_n$ in $M$. Note that $f_d$ and $f_r$ defined in this way are functions because $f$ is a function and the functional dependency holds in $T$. Also here, we can argue that $a[f(\overline{t})]$ and $a[f(\overline{t})/f_r(\overline{t}|_{\{j\}c})] \land t_j = f_d(\overline{t}|_S)$ have the same free variables $\overline{x}$ and hence the same truth value for every variable assignment $\{\overline{x}/\overline{a}\}$ under respectively $M$ and $M'$. Hence $M$ is a model of $T$ iff $M'$ is a model of $T'$. Adding the rule $\{\forall \overline{x}, x_n \in T, T_n : f(\overline{x}) = x_n \leftarrow f_r(\overline{x}|_{\{j\}c}) = x_n \land f_d(\overline{x}|_S) = x_j\}$ as prescribed by Definition 4, to $T'$, $M'$ is extended with every atom $f(\overline{a}) = d_n$ that is true in $M$ and both theories are strongly $voc(T)$-equivalent.

When $f$ is a defined symbol, every rule $f(\overline{x}) = x_n \leftarrow \varphi$ is replaced by the rules $f_d(\overline{x}|_S) = x_j \leftarrow \varphi$ and $f_r(\overline{x}|_{\{j\}c}) = x_n \leftarrow \varphi$ (Definition 5). Given the correspondence between $M$ and $M'$, clearly, an atom $f(\overline{a}) = d_n$ is supported under $M$ by the rule in $T$ iff $f_r(\overline{a}|_{\{j\}c}) = d_n$ and $f_d(\overline{a}|_S) = d_j$ are supported under $M'$ by the two rules in $T'$. Hence both theories are strongly $voc(T)$-equivalent in this case.

- The case of a functional dependency $d \langle f(T) : T_n, S, j \rangle$ with $\#(S) = n - 1$ and $j = n$ is very similar to the previous case and the proof is omitted.
- Also the case where the preprocessing step replaces atoms of the form $x = f(\overline{t})$ by $P_f(\overline{t} :: x)$ with $P_f(\overline{t} :: T_n)$ a new predicate is very similar to the above cases and we also omit its proof.

It follows that rewriting a theory $T$ according to Definitions 3, 4, and 5 produce a strongly $voc(T)$-equivalent theory. $\square$
Appendix C Handling defined functions symbols

In Example 5, the following theory, consisting of definition $\Delta$, was obtained after exploiting the functional dependencies.

\[
\begin{align*}
\begin{cases}
    f_r(s) = 0 \\
    \forall x y : f_r(x) = f_r(y) + c(y, x) \leftarrow e(y, x) \\
    \wedge \forall y' : -(y = y') \wedge e(y', x) \Rightarrow c(y, x) + f_r(y) \leq c(y', x) + f_r(y')
    \end{cases}
\end{align*}
\]

To the best of our knowledge, no search algorithm is capable of handling non-Herbrand, defined functions otherwise than by first transforming them into their predicate graph equivalent. Such a transformation increases the quantifier nesting and counteracts the aim of using functions to reduce the size of the grounding.

However, the above theory can be transformed into a theory in which the definition of $f_r$ is replaced by its completion and where a new predicate $\text{red}_{f_r}(x)$, equivalent with $\text{HasImage}(f_r(x))$ (recall, a shorthand for $\exists y : f_r(x) = y$), defines for which domain elements $f_r$ is defined. The transformation increases the size of the grounding only by a constant value.

\[
\begin{align*}
\text{comp}_{\Delta, f_r} & \quad \forall x : \text{red}_{f_r}(x) \iff \text{HasImage}(f_r(x)) \\
\text{red}_{f_r}(s) & \quad \leftarrow \\
\forall x : \text{red}_{f_r}(x) & \quad \leftarrow \exists y : \text{red}_{f_r}(y) \wedge f_r(x) = f_r(y) + c(y, x) \wedge e(y, x) \\
& \wedge (\forall y' : -(y = y') \wedge e(y', x) \wedge \text{red}_{f_r}(y') \Rightarrow c(y, x) + f_r(y) \leq c(y', x) + f_r(y'))
\end{align*}
\]

Equivalence is preserved because (i) $\text{red}_{f_r}(x)$ is total and is true iff $f_r(x)$ is defined and (ii) while there can be multiple rule instances of the original theory that define $f_r(d)$, in any model all instances with a true body will define the same value for $f_r(d)$.

Appendix D Experimental results

The detailed results on the detection and rewriting algorithm are presented in table D1. They were based on the ASP-Core-2 benchmarks of the 2013 ASP-competition, complemented with the scheduling example ("OO-Database") and the packing running example ("Packing"). For the latter, instances of the previous ASP competition were used, for the former new instances were crafted. For each entailment request, the prover was given a time limit of 2 seconds.

In Table D2, the detailed results on the model expansion experiments are shown. For 6 benchmarks, they show the average results of using model generation using a grounding to propositional $\text{PC}(\cdot)$ and using model generation with the detected functions with grounding to a richer format. A timeout of 200 seconds was used.
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#var\textsubscript{in}</th>
<th>#var\textsubscript{out}</th>
<th>#f (#tot/#part)</th>
<th>#calls</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permutation-P.-Matching</td>
<td>21</td>
<td>8</td>
<td>3(3/0)</td>
<td>18</td>
<td>18.99</td>
</tr>
<tr>
<td>Valve-Location</td>
<td>27</td>
<td>24</td>
<td>1(1/0)</td>
<td>76</td>
<td>18.99</td>
</tr>
<tr>
<td>Connected-D.-M.-Still-Life</td>
<td>31</td>
<td>31</td>
<td>0(0/0)</td>
<td>24</td>
<td>43.98</td>
</tr>
<tr>
<td>Graceful-Graphs</td>
<td>32</td>
<td>21</td>
<td>2(1/1)</td>
<td>21</td>
<td>1.49</td>
</tr>
<tr>
<td>Bottle-Filling</td>
<td>26</td>
<td>14</td>
<td>4(4/0)</td>
<td>30</td>
<td>1.49</td>
</tr>
<tr>
<td>No.Mystery</td>
<td>155</td>
<td>84</td>
<td>16(2/14)</td>
<td>181</td>
<td>363.05</td>
</tr>
<tr>
<td>Ricochet-Robots</td>
<td>109</td>
<td>77</td>
<td>13(0/13)</td>
<td>222</td>
<td>447.07</td>
</tr>
<tr>
<td>Reachability</td>
<td>5</td>
<td>5</td>
<td>0(0/0)</td>
<td>24</td>
<td>51.99</td>
</tr>
<tr>
<td>Visit-all</td>
<td>22</td>
<td>10</td>
<td>2(2/0)</td>
<td>19</td>
<td>36.99</td>
</tr>
<tr>
<td>KnightTourWithHoles</td>
<td>80</td>
<td>50</td>
<td>8(4/4)</td>
<td>63</td>
<td>118.02</td>
</tr>
<tr>
<td>Crossing-Minimization</td>
<td>38</td>
<td>28</td>
<td>1(0/1)</td>
<td>81</td>
<td>12.00</td>
</tr>
<tr>
<td>Solitaire</td>
<td>114</td>
<td>54</td>
<td>3(0/3)</td>
<td>88</td>
<td>197.05</td>
</tr>
<tr>
<td>Weighted-Sequence</td>
<td>49</td>
<td>29</td>
<td>6(6/0)</td>
<td>21</td>
<td>33.99</td>
</tr>
<tr>
<td>Stable-Marriage</td>
<td>20</td>
<td>4</td>
<td>3(0/3)</td>
<td>3</td>
<td>0.26</td>
</tr>
<tr>
<td>Incremental-Scheduling</td>
<td>89</td>
<td>60</td>
<td>4(4/0)</td>
<td>84</td>
<td>11.01</td>
</tr>
<tr>
<td>Maximal-Clique</td>
<td>7</td>
<td>7</td>
<td>0(0/0)</td>
<td>16</td>
<td>0.79</td>
</tr>
<tr>
<td>Graph-Colouring</td>
<td>9</td>
<td>4</td>
<td>1(0/0)</td>
<td>9</td>
<td>0.49</td>
</tr>
<tr>
<td>Database</td>
<td>15</td>
<td>2</td>
<td>5(5/0)</td>
<td>4</td>
<td>0.14</td>
</tr>
<tr>
<td>Packing</td>
<td>37</td>
<td>13</td>
<td>5(5/0)</td>
<td>35</td>
<td>14.01</td>
</tr>
</tbody>
</table>

Table D 1. Results on function detection and rewriting on all tested benchmarks. For each encoding, we report on the number of quantified variables in the input theory (#var\textsubscript{in}), the number that remained in the output theory (#var\textsubscript{out}). Furthermore, we report the number of introduced function symbols, separated into total and partial functions, the number of calls to the theorem prover and the total time for a complete run of the detection and rewriting algorithm.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>pred. (#)</th>
<th>func. (#)</th>
<th>pred. (sec)</th>
<th>func. (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N01-Perm.-P.-Matching</td>
<td>2/5</td>
<td>5/5</td>
<td>10</td>
<td>624</td>
</tr>
<tr>
<td>N02-ValveL.-Problem</td>
<td>1/5</td>
<td>5/5</td>
<td>800.81</td>
<td>54.47</td>
</tr>
<tr>
<td>N05-Graceful-Graphs</td>
<td>5/5</td>
<td>5/5</td>
<td>2.59</td>
<td>0.89</td>
</tr>
<tr>
<td>N20-Visit-All</td>
<td>5/5</td>
<td>5/5</td>
<td>3.18</td>
<td>2.01</td>
</tr>
<tr>
<td>OO-database</td>
<td>0/5</td>
<td>3/5</td>
<td>1000</td>
<td>415.47</td>
</tr>
<tr>
<td>Packing</td>
<td>0/5</td>
<td>5/5</td>
<td>1000</td>
<td>52.07</td>
</tr>
</tbody>
</table>

Table D 2. Results on model expansion for the benchmarks where functions were detected that are supported in the current version of MiniSAT(ID) (total functions with an integer codomain). For each, 5 instances were tested and the results report the number of instances solved for model expansion using propositional grounding (“pred”) or where the grounding can also contain functions (“func”). The last two columns indicate the total time taken.
References

