Online appendix for the paper

Detection and Exploitation of Functional Dependencies for Model Generation

published in Theory and Practice of Logic Programming

BROES DE CAT AND MAURICE BRUYNOOGHE

Department of Computer Science, KU Leuven, Belgium

(*e-mail:* {broes.decat,maurice.bruynooghe}@cs.kuleuven.be)

submitted 10 April 2013; revised 23 May 2013; accepted 23 June 2013

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 $S' \subset S$. Our transformation of $agg(\{\overline{x} \in \overline{T} : \varphi : t\})$ 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 $acc(\overline{T}) : \mathbb{N}$ as:

$$\left\{\begin{array}{ll} acc(MIN(\overline{T})) = t[\overline{x}/MIN(\overline{T})] & \leftarrow \varphi[\overline{x}/MIN(\overline{T})] \\ acc(MIN(\overline{T})) = neutral_{agg} & \leftarrow \neg\varphi[\overline{x}/MIN(\overline{T})] \\ \forall \overline{x} \in \overline{T} : acc(\overline{x}) = agg(acc(PRED(\overline{x})), t) & \leftarrow \varphi \wedge \neg(\overline{x} = MIN(\overline{T})) \\ \forall \overline{x} \in \overline{T} : acc(\overline{x}) = acc(PRED(\overline{x})) & \leftarrow \neg\varphi \wedge \neg(\overline{x} = MIN(\overline{T})) \end{array}\right\}$$

This definition is equivalent to the FO formulas

$$\begin{aligned} acc(MIN(\overline{T})) = v \equiv (v = t[\overline{x}/MIN(\overline{T})] \wedge \varphi[\overline{x}/MIN(\overline{T})]) \vee \\ (v = neutral_{agg} \wedge \neg\varphi[\overline{x}/MIN(\overline{T})]) \text{ and} \\ acc(\overline{x}) = v \equiv \neg(\overline{x} = MIN(\overline{T})) \wedge \\ (v = agg(acc(PRED(\overline{x})), t) \wedge \varphi) \vee (v = acc(PRED(\overline{x})) \wedge \neg\varphi) \end{aligned}$$

The aggregate term itself is then replaced by $acc(MAX(\overline{T}))$.

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 $l_P(\overline{T}) : \mathbb{N}$ for every defined

predicate or function symbol $P(\overline{T})$ (we only work out the predicate case in detail). This function is axiomatised to be 0 if $P(\overline{T})$ is false, and otherwise it states that $l_P(\overline{T})$ is strictly larger than the $l_Q(...)$ of atoms that were positively used to derive the truth of $P(\overline{T})$. 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 $l_P(\overline{T}) : \mathbb{N}$ also exists for all non-defined symbols $P(\overline{T})$, mapping to 0 for each tuple in $\overline{T}$.

- $\forall \overline{x} : \neg P(\overline{x}) \Rightarrow l_P(\overline{x}) = 0$ (we assume all well-founded models are total).
- $\forall \overline{x} : P(\overline{x}) \leftarrow \neg Q(\overline{x})$. The constraint is $\forall \overline{x} : P(\overline{x}) \Rightarrow l_P(\overline{x}) \geq 0$.
- $\forall \overline{x} : P(\overline{x}) \leftarrow \forall \overline{y} : Q(\overline{x}, \overline{y})$. The constraint is $\forall \overline{x}, \overline{y} : P(\overline{x}) \Rightarrow l_P(\overline{x}) > l_Q(\overline{x}, \overline{y})$.
- $\forall \overline{x} : P(\overline{x}) \leftarrow \exists \overline{y} : Q(\overline{x}, \overline{y})$. The constraint is $\forall \overline{x} : P(\overline{x}) \Rightarrow \exists \overline{y} : l_P(\overline{x}) > l_Q(\overline{x}, \overline{y})$.
- $\forall \overline{x} : P(\overline{x}) \leftarrow Q_1(\overline{x}) \vee \ldots \vee Q_n(\overline{x})$. The constraint is $\forall \overline{x} : P(\overline{x}) \Rightarrow ((Q_1(\overline{x}) \wedge l_P(\overline{x}) > l_{Q_1}(\overline{x})) \vee \ldots \vee (Q_n(\overline{x}) \wedge l_P(\overline{x}) > l_{Q_n}(\overline{x})))$.
- $\forall \overline{x} : P(\overline{x}) \leftarrow Q_1(\overline{x}) \wedge \ldots \wedge Q_n(\overline{x})$. The constraint is $\forall \overline{x} : P(\overline{x}) \Rightarrow l_P(\overline{x}) > l_{Q_1}(\overline{x}) \wedge \ldots \wedge l_P(\overline{x}) > l_{Q_n}(\overline{x})$.

The advantage of the strong FO(·)-to-FO transformation is that equivalence is preserved, and hence also all functional dependencies. However, the size of the theory increases, 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 $\mathcal{I}$ an interpretation, we denote with $P^{\mathcal{I}}$ ($f^{\mathcal{I}}$) the interpretation of predicate P (function f). Slightly abusing notation, with $\overline{d}$ a domain element, we say that $P(\overline{d}) \in \mathcal{I}$ iff the tuple $\overline{d} \in P^{\mathcal{I}}$, and similar for $f(\overline{d}) = d \in \mathcal{I}$. Given a term t with free variable $\overline{x}$ and a variable assignment $\{\overline{x}/\overline{d}\}$, we denote with $t\{\overline{x}/\overline{d}\}^{\mathcal{I}}$ the interpretation of t in $\mathcal{I}$ under that assignment (a domain element). Similar for a formula φ, $\varphi\{\overline{x}/\overline{d}\}^{\mathcal{I}}$ is the interpretation of φ 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 $\mathcal{T}$ preserve its models.

Proof
We distinguish the following operations:

- Let $\mathcal{T}'$ be derived from $\mathcal{T}$ by the replacement of a rule $\forall \overline{x} : P(\bar{t}) \leftarrow \varphi$ by $\forall \overline{x}, y : P(t_1, \ldots, t_{j-1}, y, t_{j+1}, \ldots, t_n) \leftarrow y = t_j \wedge \varphi$. Let $\mathcal{M}$ be a model of $\mathcal{T}$; we show that it is also a model of $\mathcal{T}'$. It suffices to show that every $P(\overline{d})$, element of $\mathcal{M}$ supported by the original rule in $\mathcal{T}$, is supported by the new rule in $\mathcal{T}'$. That $P(\overline{d})$ is supported means there is a variable assignment $\{\overline{x}/\overline{d}_x\}$ such that $P(\bar{t}\{\overline{x}/\overline{d}_x\}^{\mathcal{M}}) = P(\overline{d})$ and that $\varphi\{\overline{x}/\overline{d}_x\}^{\mathcal{M}} = \top$. But then $y = t_j \wedge \varphi$ is true in $\mathcal{M}$ under the variable assignment $\{\overline{x}/\overline{d}_x, y/t_j\{\overline{x}/\overline{d}_x\}^{\mathcal{M}}\}$. We can conclude from this observation that $\langle t_1, \ldots, t_{j-1}, y, t_{j+1}, \ldots, t_n\rangle \{\overline{x}/\overline{d}_x, y/t_j\{\overline{x}/\overline{d}_x\}^{\mathcal{M}}\}^{\mathcal{M}} = \bar{t}\{\overline{x}/\overline{d}_x\}^{\mathcal{M}} = \overline{d}$ 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 $\exists y : a[t/y] \wedge y = t$. Let $\mathcal{I}$ be an interpretation and let $\{\overline{x}/\overline{d}\}$ be a variable assignment for the free variables $\overline{x}$ of $a[t]$. If $a[t]\{\overline{x}/\overline{d}\}^{\mathcal{I}}$ is true then $a[t/y] \wedge y = t$ is true in $\mathcal{I}$ for the variable assignment $\{\overline{x}/\overline{d}, y/t\{\overline{x}/\overline{d}\}^I\}$ and hence also $\exists y : a[t/y] \wedge y = t$ is true in $\mathcal{I}$ 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 $\mathcal{I}$ iff $\exists y : a[t/y] \wedge y = t$ 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 $agg(\{\overline{x}, y : \varphi \wedge y = t : y\})$. Let $\{\overline{x}/\overline{d}\}$ be a variable assignment for which φ is true in I. In this case $t\{\overline{x}/\overline{d}\}^I$ is added to the set to be aggregated[1]. It follows that $\varphi \wedge y = t$ is true for exactly one variable assignment extending $\{\overline{x}/\overline{d}\}$, namely $\{\overline{x}/\overline{d}, y/t\{\overline{x}/\overline{d}\}^I\}$. Hence $y\{\overline{x}/\overline{d}, y/t\{\overline{x}/\overline{d}\}^I\}$ is added to the set to be aggregated. But $y\{\overline{x}/\overline{d}, y/t\{\overline{x}/\overline{d}\}^I\} = t\{\overline{x}/\overline{d}\}^I$ hence the same value is added. Also, for all variable assignments $\{\overline{x}/\overline{d}\}$ for which φ is false, $\varphi \wedge y = t$ 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 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 $d\langle P(\overline{T}), S, j\rangle$ holds in $\mathcal{T}$ with $\#(S) < n-1$ and $\mathcal{T}'$ is derived by replacing everywhere $P(\overline{t})$ by $P_r(\overline{t}\big|_{\{j\}^C}) \wedge t_j = f_d(\overline{t}\big|_S)$ and adding the function $f_d(\overline{T}\big|_S) : T_j$. 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 $P_r(\overline{d}\big|_{\{j\}^C})$ and $f_d(\overline{d}\big|_S) = \overline{d}\big|_{\{j\}}$ 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 $P_r(\overline{t}\big|_{\{j\}^C}) \wedge t_j = f_d(\overline{t}\big|_S)$ have the same free variables, say $\overline{x}$. Moreover, given the relationship between $\mathcal{M}$ and $\mathcal{M}'$, it holds that $P(\overline{t})\{\overline{x}/\overline{d}_x\}^{\mathcal{M}} \equiv (P_r(\overline{t}\big|_{\{j\}^C}) \wedge t_j = f_d(\overline{t}\big|_S))\{\overline{x}/\overline{d}_x\}^{\mathcal{M}'}$ for every variable assignment $\{\overline{x}/\overline{d}_x\}$ 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 $\{\forall \overline{x} \in \overline{T} : P(\overline{x}) \leftarrow P_r(\overline{x}\big|_{[1,n-1]\setminus\{j\}}) \wedge f_d(\overline{x}\big|_S) = x_j\}\}$ 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 $voc(\mathcal{T})$-equivalent.
  When P is a defined symbol, every rule $P(\overline{x}) \leftarrow \varphi$ is replaced by the rules $f_d(\overline{x}\big|_S) = x_j \leftarrow \varphi$ and $P_r(\overline{x}\big|_{\{j\}^C}) \leftarrow \varphi$ (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 $\mathcal{M}'$, clearly, an atom $P(\overline{d})$ is supported under $\mathcal{M}$ by the rule in $\mathcal{T}$ iff $P_r(\overline{d}|_{\{j\}^C})$ and $f_d(\overline{d}|_S) = d_j$ are supported under $\mathcal{M}'$ by the two rules in $\mathcal{T}'$. Hence both theories are also strongly $voc(\mathcal{T})$-equivalent in this case.

- The case of a functional dependency $d\langle P(\overline{T}), S, j\rangle$ in $\mathcal{T}$ 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 $\mathcal{T}$ with $\#(S) < n-1$, $j \neq n$, $n \notin S$, and $\mathcal{T}'$ is derived by adding the functions $f_d(\overline{T}|_S) : T_j$ and $f_r(\overline{T}|_{\{j\}^C}) : T_n$ and replacing everywhere atoms $a[f(\bar{t})]$ by $a[f(\bar{t})/f_r(\bar{t}|_{\{j\}^C})] \wedge t_j = f_d(\bar{t}|_S)$. Also here, given a total interpretation $\mathcal{M}$ of $\mathcal{T}$, we can construct a total interpretation $\mathcal{M}'$ of $\mathcal{T}'$ by copying the interpretations of all symbols but f and extending it with $f_d(\overline{d}|_S) = d_j$ and $f_r(\overline{d}|_{\{j\}^C}) = d_n$ for every atom $f(\overline{d}) = d_n$ in $\mathcal{M}$. Note that f_d and f_r defined in this way are functions because f is a function and the functional dependency holds in $\mathcal{T}$. Also here, we can argue that $a[f(\bar{t})]$ and $a[f(\bar{t})/f_r(\bar{t}|_{\{j\}^C})] \wedge t_j = f_d(\bar{t}|_S)$ have the same free variables $\overline{x}$ and hence the same truth value for every variable assignment $\{\overline{x}/\overline{d}_x\}$ under respectively $\mathcal{M}$ and $\mathcal{M}'$. Hence $\mathcal{M}$ is a model of $\mathcal{T}$ iff $\mathcal{M}'$ is a model of $\mathcal{T}'$. Adding the rule $\{\forall \overline{x}, x_n \in \overline{T}, T_n : f(\overline{x}) = x_n \leftarrow f_r(\overline{x}|_{\{j\}^C}) = x_n \wedge f_d(\overline{x}|_S) = x_j\}$ as prescribed by Definition 4, to $\mathcal{T}'$, $\mathcal{M}'$ is extended with every atom $f(\overline{d}) = d_n$ that is true in $\mathcal{M}$ and both theories are strongly $voc(\mathcal{T})$-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 $\mathcal{M}$ and $\mathcal{M}'$, clearly, an atom $f(\overline{d}) = d_n$ is supported under $\mathcal{M}$ by the rule in $\mathcal{T}$ iff $f_r(\overline{d}|_{\{j\}^C}) = d_n$ and $f_d(\overline{d}|_S) = d_j$ are supported under $\mathcal{M}'$ by the two rules in $\mathcal{T}'$. Hence both theories are strongly $voc(\mathcal{T})$-equivalent in this case.
- The case of a functional dependency $d\langle f(\overline{T}) : T_n, S, j\rangle$ 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 $x = f(\bar{t})$ by $P_f(\bar{t} :: x)$ 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($\mathcal{T}$ according to Definitions 3, 4, and 5 produce a strongly $voc(\mathcal{T})$-equivalent theory. □

Appendix C Handling defined functions symbols

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

$$\left\{\begin{array}{ll} f_r(s) = 0 & \leftarrow \\ \forall x\ y : f_r(x) = f_r(y) + c(y,x) & \leftarrow e(y,x) \\ \qquad \wedge\ \forall y' : \neg(y = y') \wedge e(y',x) \Rightarrow c(y,x) + f_r(y) \leq c(y',x) + f_r(y') & \end{array}\right\}$$

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 $red_{f_r}(x)$, equivalent with $HasImage(f_r(x))$ (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.

$$\begin{array}{l} comp_{\Delta,f_r} \\ \forall x : red_{f_r}(x) \Leftrightarrow HasImage(f_r(x)) \\ \left\{\begin{array}{ll} red_{f_r}(s) & \leftarrow \\ \forall x : red_{f_r}(x) & \leftarrow \exists y : red_{f_r}(y) \wedge f_r(x) = f_r(y) + c(y,x) \wedge e(y,x) \\ & \qquad \wedge\ (\forall y' : \neg(y = y') \wedge e(y',x) \wedge red_{f_r}(y') \\ & \qquad\qquad \Rightarrow c(y,x) + f_r(y) \leq c(y',x) + f_r(y')) \end{array}\right\} \end{array}$$

Equivalence is preserved because (i) $red_{f_r}(x)$ 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 D 1. 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 D 2, 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 PC$(\cdot)$ and using model generation with the detected functions with grounding to a richer format. A timeout of 200 seconds was used.

Benchmark	$\#var_{in}$	$\#var_{out}$	#f (#tot/#part)	#calls	time (sec)
Permutation-P.-Matching	21	8	3(3/0)	18	18.99
Valve-Location	27	24	1(1/0)	76	18.99
Connected-D.-M.-Still-Life	31	31	0(0/0)	24	43.98
Graceful-Graphs	32	21	2(1/1)	21	1.49
Bottle-Filling	26	14	4(4/0)	30	1.49
NoMystery	155	84	16(2/14)	181	363.05
Ricochet-Robots	109	77	13(0/13)	222	447.07
Reachability	5	5	0(0/0)	24	51.99
Visit-all	22	10	2(2/0)	19	36.99
KnightTourWithHoles	80	50	8(4/4)	63	118.02
Crossing-Minimization	38	28	1(0/1)	81	12.00
Solitaire	114	54	3(0/3)	88	197.05
Weighted-Sequence	49	29	6(6/0)	21	33.99
Stable-Marriage	20	4	3(0/3)	3	0.26
Incremental-Scheduling	89	60	4(4/0)	84	11.01
Maximal-Clique	7	7	0(0/0)	16	0.79
Graph-Colouring	9	4	1(0/0)	9	0.49
Database	15	2	5(5/0)	4	0.14
Packing	37	13	5(5/0)	35	14.01

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 ($\#var_{in}$), the number that remained in the output theory ($\#var_{out}$). 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.

Benchmark	pred. (#)	func. (#)	pred. (sec)	func. (sec)
N01-Perm.-P.-Matching	2/5	5/5	10	624
N02-Valve-L.-Problem	1/5	5/5	800.81	54.47
N05-Graceful-Graphs	5/5	5/5	2.59	0.89
N20-Visit-All	5/5	5/5	3.18	2.01
OO-database	0/5	3/5	1000	415.47
Packing	0/5	5/5	1000	52.07

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

Janhunen, T., Niemelä, I., and Sevalnev, M. 2009. Computing stable models via reductions to difference logic. In *LPNMR*, E. Erdem, F. Lin, and T. Schaub, Eds. LNCS, vol. 5753. Springer, 142–154.

Pelov, N. 2004. Semantics of logic programs with aggregates. Ph.D. thesis, K.U.Leuven, Leuven, Belgium.

Pelov, N. and Ternovska, E. 2005. Reducing inductive definitions to propositional satisfiability. In *ICLP*, M. Gabbrielli and G. Gupta, Eds. LNCS, vol. 3668. Springer, 221–234.