The Model Evolution Calculus and an Application to Ontological Reasoning

Peter Baumgartner

Max-Planck-Institute for Informatics

Saarbrücken,Germany

Joint work with Cesare Tinelli, U Iowa

Background – Instance Based Methods

- Model Evolution is related to Instance Based Methods
  - Ordered Semantic Hyper Linking [Plaisted et al]
  - Primal Partial Instantiation [Hooker et al]
  - Disconnection Method [Billon], DCTP [Letz&Stenz]
  - Inst-Gen [Ganzinger&Korovin]
  - First-Order DPLL [B.]
- Principle: Reduce proof search in first-order (clausal) logic to propositional logic in an „intelligent“ way
- Different to Resolution, Model Elimination,...
  (Pro‘s and Con‘s)

Background - DPLL

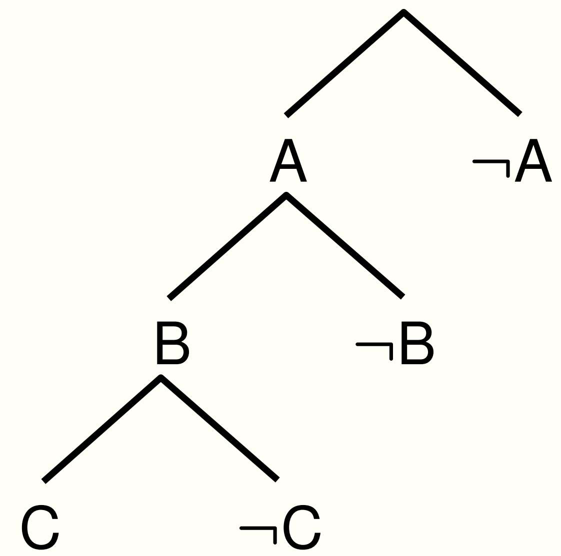
- The best modern SAT solvers (satz, MiniSat, zChaff, Berkmin,…) are based on the Davis-Putnam-Logemann-Loveland procedure [DPLL 1960-1963]
- Can DPLL be lifted to the first-order level?
  Can we combine
  - successful SAT techniques
    (unit propagation, backjumping, learning,…)
  - successful first-order techniques?
    (unification, subsumption, ...)?
- Model Evolution (ME) and its predecessor First-Order DPLL do so
- ME different to Resolution, Tableaux and Model Elimination
  but related to "Instance Based Methods"

DPLL procedure

Input: Propositional clause set
Output: Model or „unsatisfiable"

Algorithm components:

- Propositional semantic tree enumerates interpretations
- Simplification
- Split
- Backtracking



Lifting to first-order logic?

$$\{A, B\} \overset{?}{\models} \cancel{\neg A} \vee \cancel{\neg B} \vee C \vee D, \ldots$$

No, split on C: $$\{A, B, C\} \models \cancel{\neg A \vee \neg B \vee C \vee D}, \ldots$$

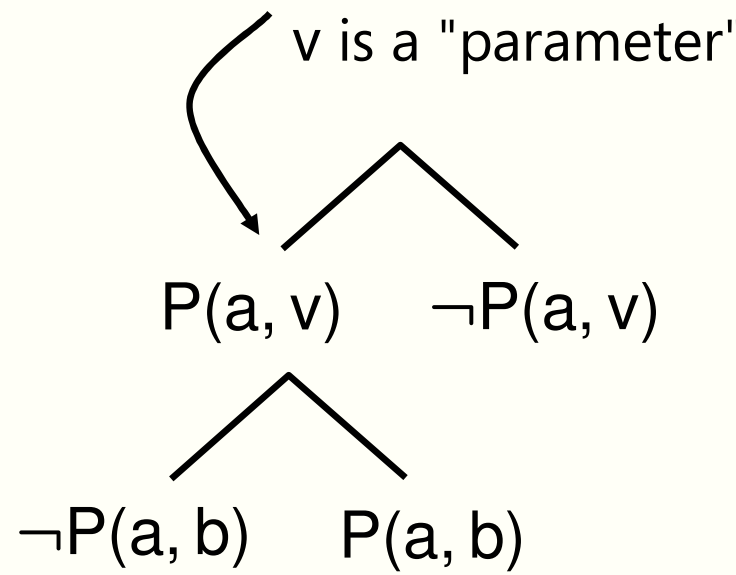
Model Evolution as First-Order DPLL

Lifing of semantic tree data structure and derivation rules to first-order

Input: First-order clause set
Output: Model or „unsatisfiable"
if termination

Algorithm components:

- First-order semantic tree enumerates interpretations
- Simplification
- Split
- Backtracking

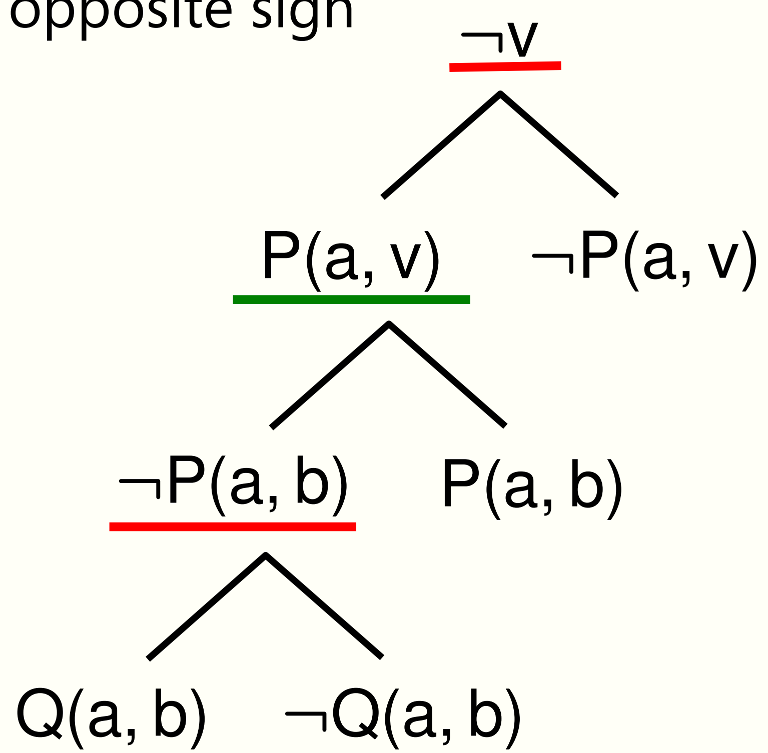


$$\{P(a, v), \neg P(a, b)\} \stackrel{?}{\models} Q(x, y) \lor P(x, y)$$

Interpretation induced by a branch?

Interpretation Induced by a Branch

A branch literal specifies the truth value of its ground instances unless there is a more specific branch literal with opposite sign



Branch:
$\{\neg v,\ P(a, z),\ \neg P(a, b)\}$

Induced Interpretation
  true: $P(a, a)$
  false: $P(a, b)$, $Q(a, b)$

How to determine Split literal?
Calculus?

$$\{\neg v, P(a, v), \neg P(a, b)\} \overset{?}{\models} Q(x, y) \vee P(x, y)$$

No, because $\{\neg v, P(a, v), \neg P(a, b)\} \not\models Q(a, b) \vee P(a, b)$

$\Rightarrow$ Split with $Q(a, b)$ to satisfy $P(a, b) \vee Q(a, b)$

Model Evolution Calculus

- Branches and clause sets may **shrink** as the derivation proceeds
- Such dynamics is best modeled with a sequent style calculus:

$$\Lambda \ \vdash \ \Phi$$

Current Clause Set

Context: A set of literals
(the „current branch“)

- **Derivation Rules**
  - To simplify the clause set Φ, to simplify the context Λ
  - Splitting
  - Close

Derivation Rules – Simplified (1)

$$\text{Split}\quad \frac{\Lambda \ \vdash\ \Phi,\ C \vee L}{\Lambda,\ L\sigma\ \vdash\ \Phi,\ C \vee L \qquad \Lambda,\ \overline{L}\sigma\ \vdash\ \Phi,\ C \vee L}$$

if

1. σ is a simultaneous mgu of $C \vee L$ against Λ,
2. neither $L\sigma$ nor $\overline{L}\sigma$ is contained in Λ, and
3. $L\sigma$ contains no variables (parameters OK)

Λ: $P(u, u)$ $Q(v, b)$

$C \vee L : \neg P(x, y) \vee \neg Q(a, z)$

$(C \vee L)\sigma : \neg P(x, x) \vee \neg Q(a, b)$

$\sigma = \{ x \mapsto u,\ y \mapsto u,\ v \mapsto a,\ z \mapsto b \}$

2. violated 2. satisfied

$L\sigma = \neg Q(a, b)$ is admissible for Split

Derivation Rules – Simplified (2)

$$\text{Close} \quad \frac{\Lambda \ \vdash \ \Phi, \ C}{\Lambda \ \vdash \ \bot}$$

if

1. $\Phi \neq \emptyset$ or $C \neq \bot$
2. there is a simultaneous mgu $\boldsymbol{\sigma}$ of C against Λ such that Λ contains the complement of each literal of $C\boldsymbol{\sigma}$

Λ: $\underline{P(u, u)}$ $\quad$ $\underline{Q(a, b)}$

$C : \neg P(x, y) \lor \neg Q(a, z)$

$$\boldsymbol{\sigma} = \{\, x \mapsto u, \ y \mapsto u, \ z \mapsto b \,\}$$

$C\boldsymbol{\sigma} : \underline{\neg P(x, x)} \lor \underline{\neg Q(a, b)}$

2. satisfied $\quad$ 2. satisfied

Close is applicable

Derivation Rules – Simplification Rules (1)

Propositional level:

$$\text{Subsume} \quad \frac{\Lambda,\ L \ \vdash \ \Phi,\ L \vee C}{\Lambda,\ L \ \vdash \ \Phi}$$

First-order level $\approx$ unit subsumption:

- All variables in context literal L must be universally quantified
- Replace equality by matching

Derivation Rules – Simplification Rules (2)

Propositional level:

Resolve
$$\frac{\Lambda,\ L \ \vdash\ \Phi,\ \overline{L} \lor C}{\Lambda,\ L \ \vdash\ \Phi,\ C}$$

First-order level ≈ restricted unit resolution

- All variables in context literal L must be universally quantified
- Replace equality by unification
- The unifier must not modify C

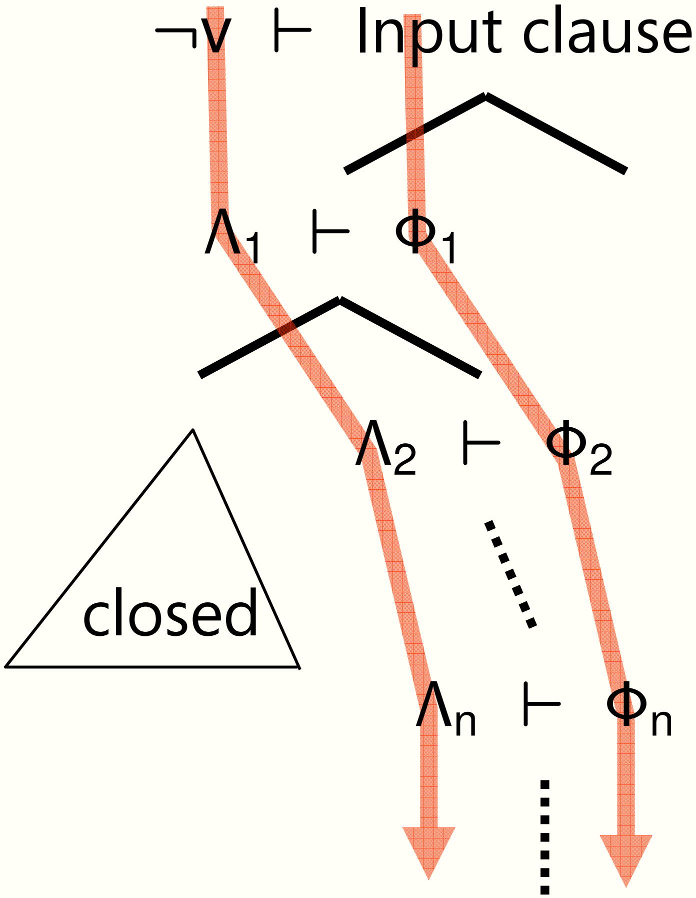
Derivation Rules – Simplification Rules (3)

$$\text{Compact} \quad \frac{\Lambda,\ K,\ L \quad \vdash \quad \Phi}{\Lambda,\ K \quad \vdash \quad \Phi}$$

if

1. all variables in K are universally quantified
2. $K\boldsymbol{\sigma} = L$, for some substitution $\boldsymbol{\sigma}$

Derivations and Completeness



$$\Lambda_\infty := \bigcup_{i \geq 0} \bigcap_{j \geq i} \Lambda_j$$

$$\Phi_\infty := \bigcup_{i \geq 0} \bigcap_{j \geq i} \Phi_j$$

Fairness

Closed tree or open limit tree, with some branch satisfying:

1. Close not applicable to any Λ_i
2. For all $C \in \Phi_\infty$ and subst. γ,
   ``if for some i, $\Lambda_i \not\models C\gamma$
   then there is $j \geq i$
   such that $\Lambda_j \models C\gamma$

(Use Split to achieve this)

Completeness

Suppose a fair derivation that is not a closed tree

Show that $\Lambda_\infty \models \Phi_\infty$

Implementation: Darwin

- „Serious“ Implementation

  Part of Master Thesis, will be continued in Ph.D. project
- (Intended) Applications
  - detecting dependent variables in CSP problems
  - strong equivalence of logic programs
  - Bernays-Schoenfinkel fragment of autoepistemic logic
- Currently extended:
  - Lemma learning
  - Equality inference rules
- Written in OCaml, 14K LOC
- User manual, proof tree output (GraphViz)

Darwin Performance

Results of ATP system competition at IJCAR 2004

MIX: Clause logic with Equality

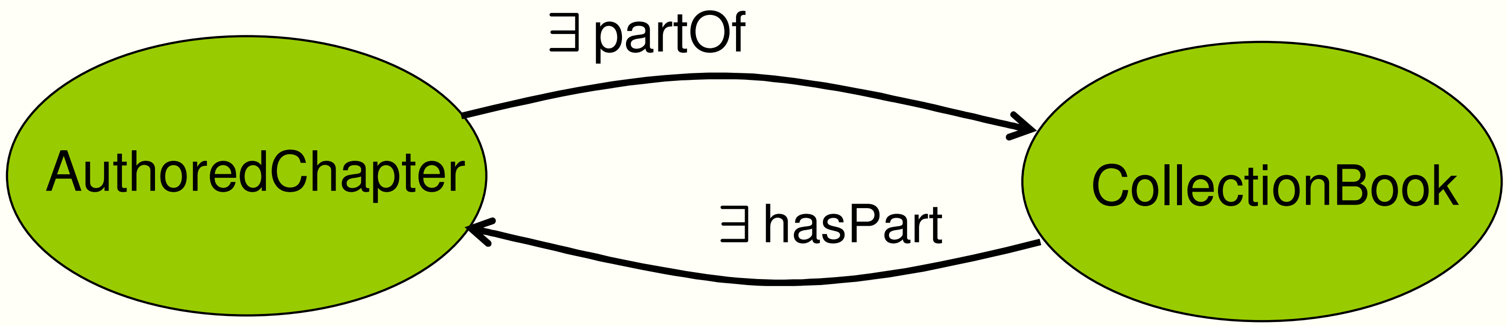
MIX	Vampire 7.0	E-SETHEO csp04	E 0.82	EP 0.82	Vampire 6.0	DCTP 10.21p	THEO J2004	DCTP 1.31	Darwin CASC-J2	SOS 1.0	Otter 3.3
Attempted	200	200	200	200	200	200	200	200	200	200	200
Solved	180	174	162	161	157	103	83	66	45	39	37
Av. Time	51.36	36.02	26.41	27.69	80.33	33.19	73.25	17.13	29.34	124.20	74.56
Solutions	180	0	0	156	157	0	82	0	0	39	37

EPR: function free clause logic (without Equality)

EPR	DCTP 10.21p	E-SETHEO csp04	Darwin CASC-J2	DCTP 1.31-EPR	DCTP 1.3-EPR	Paradox 1.1-casc	Vampire 7.0
Attempted	80	80	80	80	80	80	80
Solved	79	79	72	72	72	56	46
Av. Time	26.45	38.28	14.67	36.14	66.75	39.90	17.98
Solutions	0	0	37	0	0	28	37

Application: Ontological Reasoning

- Automated reasoning on formal ontologies is of growing interest
- Description logics are a widely used logical formalism, e.g. OWL

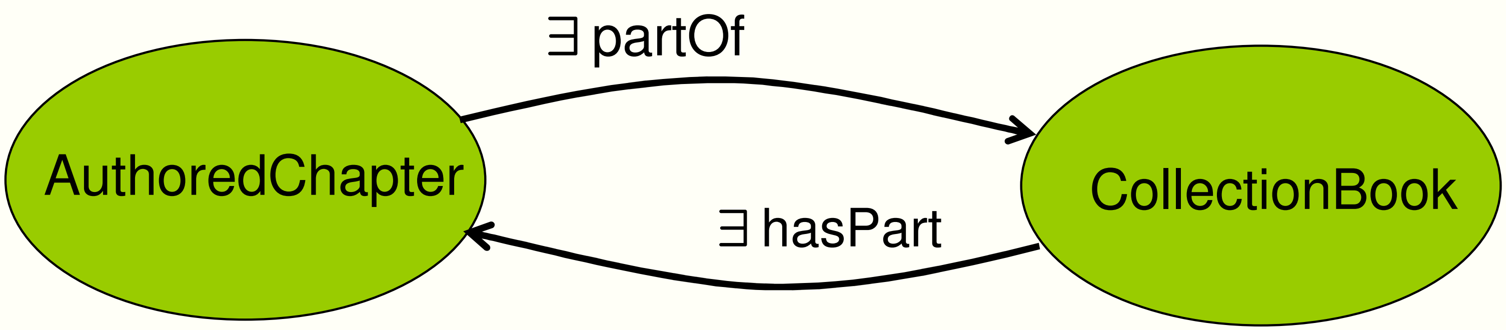


$$
\begin{aligned}
\text{AuthoredChapter} &\sqsubseteq \exists\, \text{partOf} \,.\, \text{CollectionBook} \\
\text{CollectionBook} &\sqsubseteq \exists\, \text{hasPart} \,.\, \text{AuthoredChapter}
\end{aligned}
$$

- Highly optimized reasoners for decidable DLs can cope with realistically sized ontologies (FaCT, Racer)
- **Can one also use Darwin/off-the-shelf provers?**
- **And why?**

Why? Going Beyond Description Logics

DL + Rules:



$$
\begin{aligned}
\mathrm{AuthoredChapter} &\sqsubseteq \exists\, \mathrm{partOf}\,.\,\mathrm{CollectionBook}\\
\mathrm{CollectionBook} &\sqsubseteq \exists\, \mathrm{hasPart}\,.\,\mathrm{AuthoredChapter}\\
&\ldots\\
\mathrm{EditorIsAuthor}(x) &\leftarrow \mathrm{CollectionBook}(x) \wedge \mathrm{hasPart}(x, y) \wedge\\
&\quad\ \mathrm{hasAuthor}(y, z) \wedge \mathrm{hasEditor}(x, z)
\end{aligned}
$$

- Rules cause undecidability
- Cannot use DL reasoner
- Translate to first-order logic and use theorem prover
- **How? (Naive approach fails)**

How? Our Approach

DL → **First-Order Logic** **Facts** **Rules** **Query**

Clause Normalform
Equality
Blocking

Needed for efficiency / termination

Clause Logic $\forall\,(L_1 \vee \cdots \vee L_n)$, where L_i (negated) atom

Darwin
Model Evolution

Result
- "Unsatisfiable"
- "Model"

KRHyper
Tableaux

Equality

Work in collaboration with Master's student

Equality comes in, e.g., in the translation of

- **nominals** ("oneOf")

$$\mathrm{WhiteLoire} \sqsubseteq \forall\, \mathrm{madeFromGrape} \,.\, \mathrm{Sauvignon} \sqcup \mathrm{Chenin} \sqcup \mathrm{Pinot}$$

$$\mathrm{WhiteLoire}(x) \wedge \mathrm{madeFromGrape}(x, y) \Rightarrow y = \mathrm{Sauvignon} \vee y = \mathrm{Chenin} \vee y = \mathrm{Pinot}$$

- **cardinality restrictions**

$$\mathrm{Cation} \sqsubseteq\ \leq 4\ \mathrm{hasCharge}$$

$$\mathrm{Cation}(x) \wedge \mathrm{hasCharge}(x, x1) \wedge \cdots \wedge \mathrm{hasCharge}(x, x5) \Rightarrow x1 = x2 \vee x1 = x3 \vee \cdots \vee x4 = x5$$

-> Need an (efficient) way to treat equality

Equality

- **Options**: equality axioms - builtin in prover - by transformation
- **Our transformation:**

Given clause: $c = d \quad \leftarrow \quad f(a) = b$

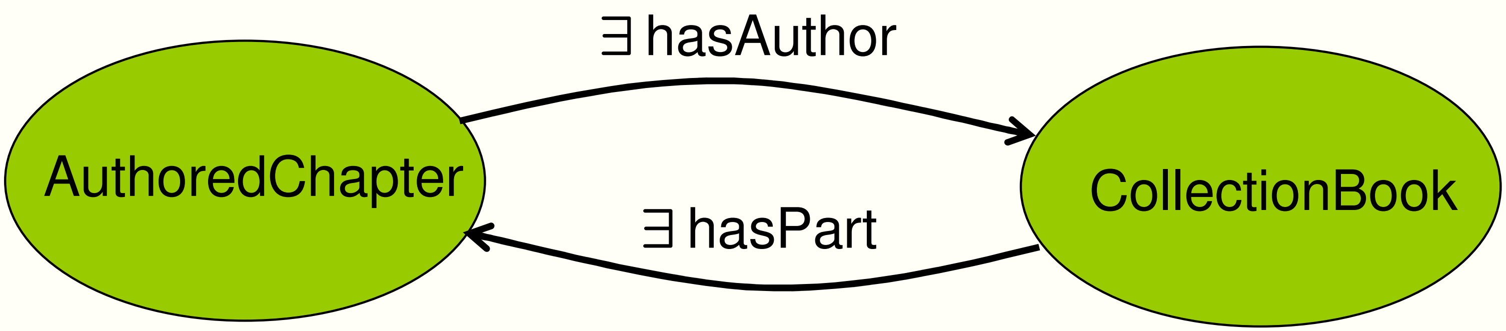
Our trafo: $c = d \quad \leftarrow \quad f(x) = b \wedge x = a$
+ ref, sym, trans

[Brand 1975]: $c = d \quad \leftarrow \quad f(x) = y \wedge x = a \wedge y = b$
$d = c \quad \leftarrow \quad f(x) = y \wedge x = a \wedge y = b$
+ ref

- Brand's transformation is theoretically more attractive
- But advantages do not apply for "typical" ontologies
- **In practice, our transformation works much better**

Blocking

- **Problem:** Termination in case of satisfiable input.
  Caused by certain DL language constructs and cyclic definitions:



- **Solution:** Idea: Re-use old individual to satisfy ∃ -quantifier.
  Technical: encode search for finite domain model in clause set:

$aC(a) \land dom(a)$

$aC(hP(hA(a))) \land hP(hA(a)) = a$ Try this first

$aC(hP(hA(a))) \land dom(hP(hA(a)))$

- **Issue:** Search space reduction: don't speculate all possible equalities

Experimental Evaluation – OWL Test Cases from W3C

System	Consistent 56 problems	Inconsistent 72 problems	Entailment 57 problems
Darwin +blocking	89%	92%	89%
Darwin - blocking	7%	94%	93%
KRHyper +blocking	86%	89%	93%
KRHyper - blocking	75%	94%	93%
Darwin ∪ KRHyper	93%	94%	93%
Hoolet (Vampire)	78%	94%	72%
Surnia (Otter)	-	0%	13%
Euler ("Prover")	0%	98%	100%
Fact (DL)	42%	85%	7%
Pellet (DL)	96%	98%	86%
OWLP (DL)	50%	26%	53%
Cerebra (DL)	90%	59%	61%
FOWL (OWL)	53%	4%	32%
ConsVISor (OWL-full)	77%	65%	-

Conclusions

- Objective: "robust" reasoning support beyond description logics:
  - Equality treatment
  - Blocking (decides standard services for cyclic ALC TBoxes)
  - It's not only "strategy hacking" – need theoretical results
  - Competitive with DL systems on common domain
- "Rules" not benchmarked (no benchmarks available), but they turned out to be very useful in own application projects:
  - Reputational risk management
  - Document management (E-Learning)
  - Upper ontology for computational linguistic application
- Nonmonotonic negation of KRHyper very useful
  How to integrate it in Model Evolution?