Reformation: A Domain-Independent Algorithm for Theory Repair

Alan Bundy
Joint work with Boris Mitrovic

School of Informatics,
University of Edinburgh

ESSENCE Lecture, Autumn School, 28^{th} October 2014

Outline

Repairing Faulty Theories

KnowItAll Ontology:

$$cap\_of(Tokyo, Japan) \qquad cap\_of(Kyoto, Japan)$$

Proof of inconsistency:

$$\dfrac{Tokyo \neq Kyoto, \quad \dfrac{cap\_of(Kyoto, Japan), \quad \dfrac{cap\_of(Tokyo, Japan), \quad cap\_of(x, z) \wedge cap\_of(y, z) \implies x = y}{cap\_of(y, Japan) \implies Tokyo = y}}{Tokyo = Kyoto}}{\Box}$$

Reformation repair:

Block unification of $cap\_of(Kyoto, Japan)$ and $cap\_of(y, Japan)$,
e.g., change $cap\_of(Kyoto, Japan)$ to $was\_cap\_of(Kyoto, Japan)$,
or add time argument to cap_of, e.g., $present$, $past$.

Repairing Planning Failures

Plan failure in ORS: Mismatch of

$$Money(PA, \pounds 200) \quad \text{and} \quad Money(PA, \pounds 200, Credit\_Card)$$

Reformation repair: Unblock failed unification.

Change planning agent's $Money/2$ to $Money/3$.

Repairing Physics Theories

Where's My Stuff Trigger:

$$\begin{aligned} O_1 &\vdash f(stuff) = v_1 \\ O_2 &\vdash f(stuff) = v_2 \\ O_{arith} &\vdash v_1 \neq v_2 \end{aligned}$$

Proof of Inconsistency:

$$\dfrac{v_1 \neq v_2, \quad \dfrac{f(stuff) = v_2, \quad \dfrac{f(stuff) = v_1, \quad y = x \wedge y = z \implies x = z}{f(stuff) = z \implies v_1 = z}}{v_1 = v_2}}{\Box}$$

Reformation Repair:

Block unification of $f(stuff) = v_2$ and $f(stuff) = z$.

e.g., rename two occurrences of *stuff* apart.

Example: Repairing a Faulty Proof of Cauchy's

Faulty Theorem: The limit of a convergent series of continuous functions is itself continuous [Cauchy].

Counter-Example: Square wave (discontinuous) is convergent sum of sine waves (continuous) [Fourier].

Failed unification:

$$y \geq y \text{ and } n \geq m(\epsilon/3, x + b(\delta(\epsilon/3, x, n)))$$

due to an occurs check failure,
where m, δ and b are Skolem function.

Repair: Change 'convergent' to 'uniformly convergent'.

Convergent:

$$\forall x. \forall \epsilon > 0. \exists m. \forall n \geq m. \, |\sum_{i=m}^{n} f_i(x)| < \epsilon$$

Uniformly Convergent:

$$\forall \epsilon > 0. \exists m. \forall x. \forall n \geq m. \, |\sum_{i=m}^{n} f_i(x)| < \epsilon$$

Note that $\forall x$ is moved to after $\exists m$.

The Standard Unification Algorithm

Case	**Before**	**Condition**	**After**
Base	$\top; \sigma$		Terminates
Trivial	$s \equiv s \wedge u; \sigma$		$u; \sigma$
Decomp	$f(\vec{s}^n) \equiv f(\vec{t}^n) \wedge u; \sigma$		$\bigwedge_{i=1}^n s_i \equiv t_i \wedge u; \sigma$
Clash	$f(\vec{s}^m) \equiv g(\vec{t}^n) \wedge u; \sigma$	$f \neq g \vee m \neq n$	fail
Orient	$t \equiv x \wedge u; \sigma$		$x \equiv t \wedge u; \sigma$
Occurs	$x \equiv s \wedge u; \sigma$	$x \in \mathcal{V}(s) \wedge x \neq s$	fail
Var Elim	$x \equiv s \wedge u; \sigma$	$x \notin \mathcal{V}(s)$	$u\{x/s\}; \sigma \oplus \{x/s\}$

- Adapted from [Baader & Snyder, 2001][p455].
- Returns unique most-general unifier.

The Modified Unification Algorithm

Case	Before	Condition	After
Base	$\top; \sigma$		Terminates
CC_s	$f(\vec{s}^m) \equiv g(\vec{t}^n)$	$f = g \wedge n = m$	$\bigwedge_{i=1}^{n} s_i \equiv t_i \wedge u; \sigma$
CC_f	$\wedge u; \sigma$	$f \neq g \vee n \neq m$	Fail
VC_f	$x \equiv t \wedge u; \sigma$	$x \in \mathcal{V}(t)$	Fail
VC_s	or $t \equiv x \wedge u; \sigma$	$x \notin \mathcal{V}(t)$	$u\{x/t\}; \sigma \oplus \{x/t\}$
$VV_=$	$x \equiv x \wedge u; \sigma$		$u; \sigma$
$VV_{\neq}$	$x \equiv y \wedge u; \sigma$	$x \neq y$	$u\{x/y\}; \sigma \oplus \{x/y\}$

- Equivalent to standard unification algorithm.
- Groups compound/compound and variable/compound cases into success/fail.

The Reformation Algorithm

Case	Before	Condition	Block	Unblock
Base	$\top$		Failure	Success
CC_s	$f(\vec{s}^m) \equiv g(\vec{t}^n) \wedge u$	$f = g \wedge m = n$	Make $f(\vec{s}^m) \neq f(\vec{t}^m)$ $\bigvee_{i=1}^n$ Block $s_i \equiv t_i$ $\vee$ Block u	$\bigwedge_{i=1}^n$ Unblock $s_i \equiv t_i$ $\wedge$ Unblock u
CC_f	$f(\vec{s}^m) \equiv g(\vec{t}^n) \wedge u$	$f \neq g \vee m \neq n$	Success	Make $f(\vec{s}^m) = g(\vec{t}^n)$ $\bigwedge_{i=1}^n$ Unblock $\nu(s_i) \equiv \nu(t_i)$ $\wedge$ Unblock $\nu(u)$
VC_f	$x \equiv t \wedge u$ or $t \equiv x \wedge u$	$x \in \mathcal{V}(t)$	Success	Make $x \notin \mathcal{V}(t)$ $\wedge$ Unblock $\nu(u\{x/t\})$
VC_s	$x \equiv t \wedge u$ or $t \equiv x \wedge u$	$x \notin \mathcal{V}(t)$	Make $x \in \mathcal{V}(t)$ $\vee$ Block $u\{x/t\}$	Unblock $u\{x/t\}$

- Adapts modified unification algorithm.
- Flips success and failure cases to block/unblock unification.
- Blocking is a disjunction; unblocking a conjunction.
- Implemented and evaluated in SWI Prolog.

Example: Family Relations

Unprovable truth:

$$\frac{\mathit{Mother}(\mathit{Camilla}, \mathit{William}), \quad \neg \mathit{Mother}(p, c, \mathit{Step}) \vee \mathit{StepMother}(p, c)}{\times}$$

1 of 2 repairs: Add 3^{rd} argument to $\mathit{Mother}(\mathit{Camilla}, \mathit{William})$.

Successful resolution:

$$\frac{\mathit{Mother}(\mathit{Camilla}, \mathit{William}, \mathit{Step}), \quad \neg \mathit{Mother}(p, c, \mathit{Step}) \vee \mathit{StepMother}(p, c)}{\mathit{StepMother}(\mathit{Camilla}, \mathit{William})}$$

The Many-Sorted Reformation Algorithm

Case	Before	Condition	Block	Unblock
Base	$\top$		Failure	Success
CC_s	$f(\vec{s}^m){:}\tau_f \equiv g(\vec{t}^n){:}\tau_g \wedge u$	$f = g \wedge m = n$	Make $f(\vec{s}^m) \neq f(\vec{t}^m)$ $\bigvee_{i=1}^n$ Block $s_i \equiv t_i$ $\vee$ Block u	$\bigwedge_{i=1}^n$ Unblock $s_i \equiv t_i$ $\wedge$ Unblock u
CC_f		$f \neq g \vee m \neq n$	Success	Make $f(\vec{s}^m) = g(\vec{t}^n)$ $\bigwedge_{i=1}^n$ Unblock $s_i \equiv t_i$ $\wedge$ Unblock u
VC_s	$x{:}\tau_x \equiv t{:}\tau_t \wedge u$ or $t{:}\tau_t \equiv x{:}\tau_x \wedge u$	$x \notin V(t) \wedge \tau_t \preceq^* \tau_x$	Make $x \in V(t)$ $\vee \tau_t \not\preceq^* \tau_x$ $\vee$ Block $u\{x{:}\tau_x / t{:}\tau_t\}$	Unblock $u\{x{:}\tau_x / t{:}\tau_t\}$
VC_f		$x \in V(t) \vee \tau_t \not\preceq^* \tau_x$	Success	Make $x \notin V(t)$ $\wedge \tau_t \preceq^* \tau_x \wedge$ Unblock $u\{x{:}\tau_x / t{:}\tau_t\}$
VV_s	$x{:}\tau_x \equiv y{:}\tau_y \wedge u$	$D = glbs(\tau_x, \tau_y) \wedge D \neq \emptyset$	Make $glbs(\tau_x, \tau_y) = \emptyset$ $\vee$ $\bigwedge_{\tau_d \in D}$ Block $u\{x{:}\tau_x / y{:}\tau_d, y{:}\tau_y / y{:}\tau_d\}$	$\bigvee_{\tau_d \in D}$ Unblock $u\{x{:}\tau_x / y{:}\tau_d, y{:}\tau_y / y{:}\tau_d\}$
VV_f		$glbs(\tau_x, \tau_y) = \emptyset$	Success	Make $glbs(\tau_x, \tau_y) \neq \emptyset$ $\wedge$ Unblock $u\{x{:}\tau_d / y{:}\tau_d\}$

- Extended reformation to many-sorted logics.
  - Repairs now include splitting and merging of sorts.
  - Plus reorganisation of sort hierarchy.

Example: Flying Penguins

Contradiction:

$$\frac{\neg Flies(Penguin4 : Penguin), \quad Flies(x : FlyingAnimal)}{\Box}$$

where $Penguin \prec Bird$ and $Bird \prec FlyingAnimal$.

1 of 3 repairs: Split $Bird$ into $FlyingBird$ and $Bird$.

- Replace $Bird \prec FlyingAnimal$ with $FlyingBird \prec FlyingAnimal$.
- Add $FlyingBird \prec Bird$.

Search Space Control

- Huge search space: many possible repairs for every unwanted unification.
  - Each proof step requires unification.
  - Each unification step suggests multiple repairs.
- Need heuristics to prune and prioritise.
  - Protect some functions/predicates.
  - Keep repairs minimal.
  - Maximise blocked inconsistencies; minimise blocked truths.

Conclusion

- Language repair essential in many applications.
- Reformation is general-purpose algorithm.
- Huge search space requires heuristic control.
- Need to define *minimality*.
- Explore extensions to other logics, e.g., DL.

Baader, F. and Snyder, W.
(2001).
Unification theory.
In Robinson, J. A. and Voronkov, A., (eds.), *Handbook of Automated Reasoning, Volume 1*, volume I, chapter 8, pages 447–553. Elsevier.