Experiments With Connection Method Provers

Wolfgang Bibel
Emeritus DUT & UBC
Jens Otten
University of Oslo

Plan for talk

- Is ATP part of the current AI hype?
- The historical role of the connection method (CM) within Logic and ATP
- Features, calculi and systems of the CM
- Clausal vs. non-clausal CM
- Need for more intelligence & deep learning in ATP systems
- Conclusions

AI and Automated Deduction

- AI revolutionized understanding of intelligent behaviour – resulting in
  - autonomous vehicles; worldmasters in chess, Go, poker, Jeopardy!, StarCraft; first proofs of deep mathematical theorems; countless applicational systems
- Fact: still side role of AD in AI
- Two possible reasons: 1. irrelevant? **No!**

AD‘s Crucial Role in AI

- Intelligent agents sense the environment, take actions based on world model which is learned, inductively inferred and deduced
- Great successes with deep learning
- No intelligence without additionally acquired knowledge hence deductive/ inductive inference remains crucial

Most Likely Second Reason

- Only other reason for AD‘s side role:
  AD has not yet reached the necessary level of performance and useability
- **Why?**

- Talk will try to give some answers and thereby provide a vision for the future
- Let us start with a short history of AD

H-Systems, G-Systems, CM

- Herbrand‘s interest in *finding* proofs
- H-systems based on Herbrand‘s theorem (1929) resulting in
  - Resolution and its early successes
- Gentzen systems modelling reasoning
- G-systems, like eg. tableaux
- CM extremely compressed version of tableaux, hence is G-system as well

Detailed Plan for Talk

- CM‘s formula-orientedness involving connections & unification resulting in
  - Compactness and high performance
  - Uniformity over many logics
  - Global view over the object of analysis
- Structure of talk determined by these three features unique for CM
- Culminating in vision for future AD

First example F_1

$$\exists a (Pa \vee \neg \exists x\, Qx) \rightarrow \exists y\, Py \vee \forall b\, \neg Q f b$$

$$\sigma = \{x_1 \backslash fb, y_1 \backslash a\}$$

A connection proof of formula F_1

Gentzen Schütte Tableaux CM

- Gentzen sequent calculus with 19 rules
- Schütte‵s generative formal system GS with $\neg$, $\vee$, $\exists$ and 3 rules of inference, already a substantial simplification
- Beth‵s tableaux much like GS, but analytic and proof by contradiction of negated formula
- CM a compressed version of GS

Derivation of F_1 in GS vs CM

$$
\dfrac{
  \dfrac{
    \dfrac{\neg\mathbf{Pa} \vee \exists y\, Py \vee \mathbf{Pa} \vee \neg\exists b\, Qfb}{\neg Pa \vee \exists y\, Py \vee \neg\exists b\, Qfb}\,\exists
    \qquad
    \dfrac{\dfrac{\neg\neg(\exists x\, Qx \vee \mathbf{Qfb}) \vee \exists y\, Py \vee \neg\mathbf{Qfb}}{\neg\neg\exists x\, Qx \vee \exists y\, Py \vee \neg Qfb}\,\exists}{\neg\neg\exists x\, Qx \vee \exists y\, Py \vee \neg\exists b\, Qfb}\,\forall
  }{\neg(Pa \vee \neg\exists x\, Qx) \vee \exists y\, Py \vee \neg\exists b\, Qfb}\,\wedge
}{\neg\exists a\,(Pa \vee \neg\exists x\, Qx) \vee \exists y\, Py \vee \neg\exists b\, Qfb}\,\forall
$$

$$\neg\exists a\,(P^1 a \vee \neg\exists x\, Q^0 x) \vee \exists y\, P^0 y \vee \neg Q^1 fb \quad \text{with} \quad \sigma = \{x_1 \backslash fb, y_1 \backslash a\}$$

Connection Proofs

- Connection proofs are derivations in Gentzen's formal system LK reduced to their very essence by eliminating all redundancies from them
- Transformation between the two representations easily realizable
- Hence ease for interaction with humans

CM vs Tableaux

- Connection proof much more compact
- Redundancy removed, connection-guided
- Hence much higher performance as
  - demonstrated in CASC competitions
- All other virtues of tableaux inherited
- Thus if performance counts then the CM is the method of choice in comparison with tableaux, GS etc.

CM vs Resolution

- Eder has shown in 1993 that a more refined version of the CM, the connection structure calculus, can linearly simulate any resolution proof
- Thus in this sense the CM is at least as powerful as resolution as well
- Has partially been implemented in SETHEO, but not yet in any leanCoP

Matrix Representation

$$\left[\begin{bmatrix} \neg Pa \\ Qx \end{bmatrix} \begin{bmatrix} Py \end{bmatrix} \begin{bmatrix} \neg Qfb \end{bmatrix} \right] \qquad \sigma = \{x_1 \backslash fb, y_1 \backslash a\}$$

$$\left[\begin{bmatrix} \neg N0 \end{bmatrix} \begin{bmatrix} Nx_1 \\ \neg Nfx_1 \end{bmatrix} \begin{bmatrix} \neg Nfx_2 \\ Nx_2 \end{bmatrix} \begin{bmatrix} Nff0 \end{bmatrix} \right] \qquad \sigma = \{x_1 \backslash 0, x_2 \backslash f0\}$$

Number formula F_2

2 instances of number formula

$$N0 \wedge \forall x(Nx \rightarrow Nfx) \rightarrow Nff0 \quad \text{with} \quad \sigma = \{x_1 \backslash 0, x_2 \backslash f0\}$$

n instances of rule in number formula

$$N0 \wedge \forall x(Nx \rightarrow Nfx) \rightarrow Nf^n0 \quad \sigma = \{x_1 \backslash 0,\ x_{i+1} \backslash fx_i, i = 1, \ldots, n-1\}$$

History of Connection Proofs

n instances of rule in number formula

$$N0 \wedge \forall x(Nx \rightarrow Nfx) \rightarrow Nf^n0 \qquad \sigma = \{x_1 \backslash 0,\ x_{i+1} \backslash fx_i, i = 1, \ldots, n-1\}$$

First connection proof in Habil-thesis 1974

6.5. $\neg fac(0)=1 \vee \exists x \exists y(fac(x)=y \wedge \neg fac(x+1)=y \cdot (x+1)) \vee \forall x \exists y\ fac(x)=y$

$i=1,2,\ldots$

Unification: Ordering Approach

$$\forall b(\forall a \exists z(\exists x\, Paxz \wedge_1 \exists y\, Pbyz) \vee \forall c \exists u \neg Puuc \wedge_2 \forall d \exists v \neg Pvbd)$$

$$\wedge_2 <\cdot\, z,\ \wedge_1 <\cdot\, u \text{ and } \wedge_1 <\cdot\, v$$

$$\sigma = \{x \backslash a, x' \backslash b, y \backslash a, y' \backslash b, z \backslash c, z' \backslash d, u \backslash a, u' \backslash b, v \backslash a, v' \backslash b\}$$

Modal and Higher-Order Logic

$$\Box\, man(Plato) \rightarrow \Diamond\, man(Plato)$$

$$\forall ab \forall P \exists X \left[(Xa \rightarrow Xb) \rightarrow (Pb \rightarrow Pa) \right]$$

$$\sigma = \{X \backslash \lambda z \neg Pz\}.$$

Same for Many Logics

- Modal and Intuitionistic Logic require prefixes and their additional unification
- Thus again connections & unification, ie. uniformity, thanks to Jens Otten et al.
- Systems nanoCoP[-i/-M], MleanCoP and ileanCoP with highest performances
- Could as well be realized by a further generalization of the ordering approach

Connection Calculi

- Search for subset U of connections s.t.
  - unifiable (mostly fast)
  - spanning (hard part)
- Two basic principles
  - If A $\rightarrow$ D then start with connections in D
  - If connection in U hits a clause then all ist literals are involved in U
- Numerous refinements in literature

Jens Otten`s Prover leanCoP 2.0

```
prove(I,S) :- \+member(scut,S) -> prove([-(#)],[],I,[],S) ;
    lit(#,C,_) -> prove(C,[-(#)],I,[],S).
prove(I,S) :- member(comp(L),S), I=L -> prove(1,[]) ;
    (member(comp(_),S);retract(p)) -> J is I+1, prove(J,S).
prove([],_,_,_,_).
prove([L|C],P,I,Q,S) :- \+ (member(A,[L|C]), member(B,P),
    A==B), (-N=L;-L=N) -> ( member(D,Q), L==D ;
    member(E,P), unify_with_occurs_check(E,N) ; lit(N,F,H),
    (H=g -> true ; length(P,K), K<I -> true ;
    \+p -> assert(p), fail), prove(F,[L|P],I,Q,S) ),
    (member(cut,S) -> ! ; true), prove(C,P,I,[L|Q],S).
```

A Fair Question

- If a four-clauses program in high-level (and thus relatively inefficient) PROLOG can favorably compete with programs consisting of hundreds of thousands lines of code in efficient low-level languages like C++
  - what does this say about the underlying proof **methods** used in those programs?

Features of Connection Calculi

- Formula-oriented
- Uniformly covering many logics
- Goal-oriented, connection-guided
- Many enhancements in detail such as restricted backtracking and others
- Overall: CM unique & unrivalled
- Global view on – possibly very large – formulas

Clausal vs Non-Clausal CM

- Nearly all TPs employ clausal form
- nanoCoP non-clausal (formula-oriented)
- Question: non-clausal worthwhile?
- Extensive experimental comparison of leanCoP vs nanoCoP performance on 7151 FOF problems in TPTP library, for each of its 40 domains separately
- Both provers in adapted core versions

Illustrative Example

- Example: $(A \Rightarrow A) \wedge (B \Rightarrow B) \wedge (C \Rightarrow C)$

1. Standard translation:
   $(A \wedge B \wedge C) \vee (A \wedge B \wedge \neg C) \vee \ldots \vee (\neg A \wedge \neg B \wedge \neg C)$
   $$\begin{bmatrix} A \\ B \\ C \end{bmatrix} \begin{bmatrix} A \\ B \\ \neg C \end{bmatrix} \begin{bmatrix} A \\ \neg B \\ C \end{bmatrix} \begin{bmatrix} A \\ \neg B \\ \neg C \end{bmatrix} \begin{bmatrix} \neg A \\ B \\ C \end{bmatrix} \begin{bmatrix} \neg A \\ B \\ \neg C \end{bmatrix} \begin{bmatrix} \neg A \\ \neg B \\ C \end{bmatrix} \begin{bmatrix} \neg A \\ \neg B \\ \neg C \end{bmatrix}$$
   33 proof steps for connection, sequent or tableau proof

2. Definitional translation:
   $((A \Rightarrow A) \Rightarrow P) \wedge (B \Rightarrow B) \Rightarrow Q) \wedge (C \Rightarrow C) \Rightarrow R)) \Rightarrow (P \wedge Q \wedge R)$
   $$\begin{bmatrix} P \\ Q \\ R \end{bmatrix} \begin{bmatrix} \neg P \\ A \end{bmatrix} \begin{bmatrix} \neg P \\ \neg A \end{bmatrix} \begin{bmatrix} \neg Q \\ B \end{bmatrix} \begin{bmatrix} \neg Q \\ \neg B \end{bmatrix} \begin{bmatrix} \neg R \\ C \end{bmatrix} \begin{bmatrix} \neg R \\ \neg C \end{bmatrix}$$
   9 proof steps

3. Non-clausal: $(A \Rightarrow A) \wedge (B \Rightarrow B) \wedge (C \Rightarrow C)$
   $$\begin{bmatrix} [A \quad \neg A] \\ [B \quad \neg B] \\ [C \quad \neg C] \end{bmatrix}$$
   3 proof steps

Results on „non-clausal“ probl.

	#proved problems			nanoCoP		average proof time		average proof size	
Domain	leanCoP	nanoCoP	both	time$\leq$	size$<$	leanCoP	nanoCoP	leanCoP	nanoCoP
AGT	18	18	18	11	0	0.2	0.4	6	6
ALG	6	10	4	3	3	0.5	0.7	1031	140
BIO	0	0	0	–	–	–	–	–	–
BOO	0	0	0	–	–	–	–	–	–
CAT	2	2	2	2	0	0.3	0.2	75	75
COM	2	9	2	2	0	0.1	0.1	4	4
CSR	40	37	37	4	0	0.6	1.2	24	25
GEG	0	0	0	–	–	–	–	–	–
GEO	186	189	185	98	19	0.5	0.5	13	13
GRA	4	3	3	3	3	1.8	1.6	27	24
GRP	2	4	2	1	1	0.6	0.6	72	60
HAL	1	1	1	0	1	3.2	5.2	11	7
HWV	0	0	0	–	–	–	–	–	–
KLE	3	3	3	1	0	0.2	0.6	7	7
KRS	83	89	81	48	64	0.4	0.5	33	17
LAT	10	12	10	4	4	0.9	0.6	25	23
LCL	45	41	41	22	19	0.4	0.6	11	8
MED	0	5	0	–	–	–	–	–	–
MGT	26	27	24	12	10	0.4	0.6	24	20
MSC	2	2	2	0	1	0.2	0.4	30	30
NLP	3	15	3	3	3	0.3	0.1	687	32

Bottom Line of Experiment

- For clausal problems no advantage
- For inherently non-clausal problems nanoCoP proves more problems with significantly shorter proofs
- Eg. NLP117+1: 782 vs 34 connections
- **Note**:
  some really deep problems will thus be provable by a non-clausal prover only

Global Aspects

- Abbreviating the antecedent in

$$N0 \wedge \forall x(Nx \rightarrow Nfx) \rightarrow Nf^n0 \qquad \sigma = \{x_1\backslash 0,\ x_{i+1}\backslash fx_i, i = 1,\ldots,n-1\}$$

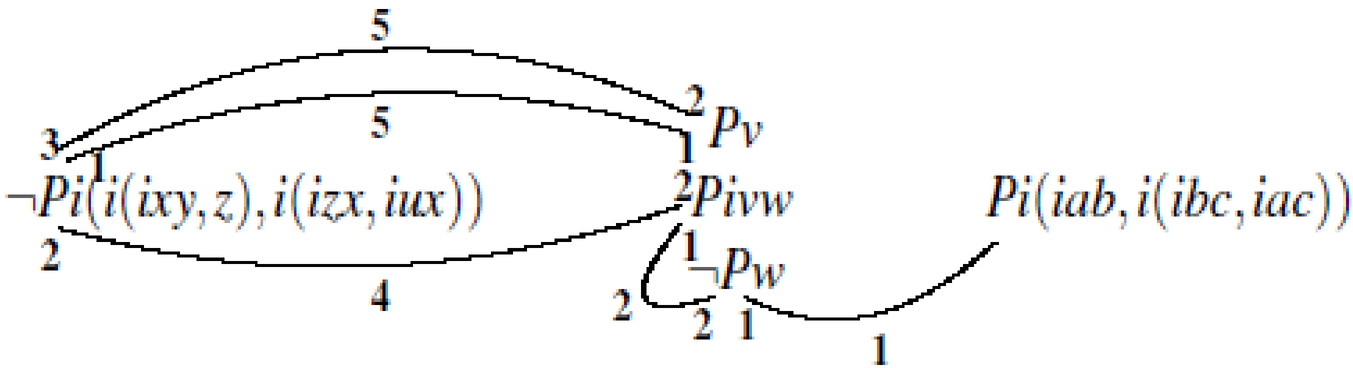
by Nf^z0 reduces proof search to a single connection and unification of z with n

- Many more opportunities of this kind in the literature, current focus is mainly on speed rather than more intelligence

Łukasiewicz with Connections

$$Pi(i(ixy,z),i(izx,iux)) \wedge (Pv \wedge Pivw \rightarrow Pw) \rightarrow Pi(iab,i(ibc,iac))$$

Two Rule Applications



$$
\begin{aligned}
\sigma_2 = \{ & x_1 \backslash x_1, \ldots, u_1 \backslash u_1, v_1 \backslash i(i(icy_1,b), i(ibc,iac)), w_1 \backslash i(iab, i(ibc,iac)), \\
& x_2 \backslash i(ibc,iac), y_2 \backslash z_3, z_2 \backslash i(icy_1,b), z_2 \backslash i(i(z_3,ibc), i(u_1,ibc)), \\
& u_2 \backslash iab, v_2 \backslash i(i(i(ibc),(iac),y_2), i(i(y_2,ibc), i(u_3,ibc))), \\
& w_2 \backslash i(i(i(icy_1,b), i(ibc,iac)), i(iab, i(ibc,iac))), x_3 \backslash ibc, y_3 \backslash iac, z_3 \backslash z_3, \\
& u_3 \backslash u_3 \}
\end{aligned}
$$

Łukasiewicz Example

- Features 5 basic unifiable connections
- Find sequence of connection instances
- Łukasiewicz found 29 steps proof
- Systems need 3.3k to 7m search steps
- Deep learning selecting connections (states characterized by substitutions)
- Crude speed a weak counter argument

Global Aspects of Proof Search

- Abbreviation technique for abounding recursive features in problems
- Deep learning techniques for cycle problems
- Same for large theories in order to learn a „feeling" which theorems apply to the given problem
- Global view of CM: mathematicians

Conclusions and Vision

- CM is method of choice due to compactness/performance + uniformity + global view vs. tableaux & resolution
- Extreme intellectual challenge
- Numerous features of detail known but never integrated in any system
- Potential of deep learning for CM
- Time to initiate an international project!

An Urgent Call

In order to solve the world‘s extremely complex problems endangering the future existence of mankind (like global warming etc.) we urgently need more rationality in problem solving. Given the nature of humans, only rationality built into *artificially intelligent rational agents* (AIRAs) are likely to save us from desaster. AD will be a crucial part thereby.

Advertisements

New Books   for German language readers

L. Wolfgang Bibel, *Reflexionen vor Reflexen – Memoiren eines Forschers* Cuvillier Verlag, Göttingen, 2017

W. Bibel & U. Furbach, *Formierung eines Forschungsgebiets* Preprint 15, Dt. Museum Verlag, München, 2018

Fresh Perspectives for KR

- Take formulas and connections as basis
- Default reasoning realised by way of
  - preference among sets of connections (eg. according simplicity, learned weights)
- Fuzzy/probabilistic reasoning by
  - Connections with weights attached
- See early papers of 1980‘s by author

Connection Method (CM)

- Proving a formula F means eg. finding a proof in a Gentzen-type formal system
- Compression principle: find minimal essentials of a proof, called *skeletons* :
- multiplicity, spanning set of connect-ions, partial ordering, substitution
- Search for skeleton on F in a goal- and connection-oriented, by-need fashion

Why better than resolution …?

- Search space consisting of
  - small skeletons rather than possibly huge derivations, an obvious advantage speeding up any necessary operations
  - search more driven by given structures
  - each skeleton represents a number of derivations which differ in irrelevant and redundant features
- … but the cut is missing … see below

Otten‘s theorem prover for Intuitionistic Logic: ileanCoP

```
(1) prove(Mat,PathLim) :-
(2) append(MatA,[FV:Cla|MatB],Mat), \+ member(-(_ ):_ ,Cla),
(3) append(MatA,MatB,Mat1),
(4) prove([!:[]],[FV:[-(!):(-[])|Cla]|Mat1],[],PathLim,[PreSet,FreeV]),
(5) check_addco(FreeV), prefix_ unify(PreSet).
(6) prove(Mat,PathLim) :-
(7) \+ ground(Mat), PathLim1 is PathLim+1, prove(Mat,PathLim1).
(8) prove([],_,_,_,[[],[]]).
(9) prove([Lit:Pre|Cla],Mat,Path,PathLim,[PreSet,FreeV]) :-
(10) (-NegLit=Lit;-Lit=NegLit) ->
(11) ( member(NegL:PreN,Path), unify_ with_ occurs_
     check(NegL,NegLit),
(12) \+ \+ prefix_ unify([Pre=PreN]), PreSet1=[], FreeV3=[]
(13) ;
(14) append(MatA,[Cla1|MatB],Mat), copy_ term(Cla1,FV:Cla2),
(15) append(ClaA,[NegL:PreN|ClaB],Cla2),
```

Rest of ileanCoP without unification – leanCoP included

```
(16) unify_ with_ occurs_ check(NegL,NegLit),
(17) \+ \+ prefix_ unify([Pre=PreN]),
(18) append(ClaA,ClaB,Cla3),
(19) ( Cla1==FV:Cla2 ->
(20) append(MatB,MatA,Mat1)
(21) ;
(22) length(Path,K), K<PathLim,
(23) append(MatB,[Cla1|MatA],Mat1)
(24) ),
(25) prove(Cla3,Mat1,[Lit:Pre|Path],PathLim,[PreSet1,FreeV1]),
(26) append(FreeV1,FV,FreeV3)
(27) ),
(28) prove(Cla,Mat,Path,PathLim,[PreSet2,FreeV2]),
(29) append([Pre=PreN|PreSet1],PreSet2,PreSet),
(30) append(FreeV2,FreeV3,FreeV).
```

First Challenge

- 3 clauses, leanCoP 333 bytes, ileanCoP additional 191 bytes in smallest versions
  http://en.wikipedia.org/wiki/Automated_theorem_proving
- Integrate full power of partial relation (as in Bibel ATP book) and preprocess F by applying reduction operations
- Transformation to lower-level programming language, eg. C++, like in Mercury

Second Challenge: Cut

- Cut enables exponential compression
- *Conjecture*: disappears by eliminating common factors in different clauses
- Integrate FACTOR-reduction in leanCoP
- Would overcome the remaining advant-age of resolution in comparison with CM
- Evidences: Letz‘ folding-up in SETHEO; pigeon-hole formulas; redundancy elim.

Third Challenge: Dynamics

- Logic a framework for static reasoning
- Ubiquitous need to cope for changes
- Problems with previous attemps
- Transition calculus in new form incorporates transitions as first-class citizens without frame problem
- Integrate in leanCoP