Explanations in Constraint Programming

Barry O'Sullivan

Cork Constraint Computation Centre
Department of Computer Science
University College of Cork, Ireland

email: `b.osullivan@cs.ucc.ie`

ACP Summer School 2008

Why do we care about Explanations?

Configuration as a CSP

- A "product" is fully specified by some constraints
- Several options are available to the user
- The user expresses his preferences as constraints

Explanations

When preferences conflict:

Conflict show a set of conflicting preferences

Relaxation show a set of feasible preferences

Why do we care about Explanations?

Debugging a CSP Model

- A model represents a reality using some constraints
- The programmer "proposes" a model

Explanations

When the model/reality conflict:

Conflict show a set of conflicts between the model and reality

Relaxation show a set of feasible constraints

Classic Setting

Two Categories of Constraints

- *background constraints* expressing the connections between the components of the “product”, that cannot be removed
- *user constraints* interactively stated by the user when deciding on options (= a query)

Consistency

- A set of constraints is *consistent* if it admits a solution.
- The background constraints are assumed to be consistent.
- The “solubility” of a set of constraints refers to the number of solutions it is consistent with.

Terminology

Explanations

- **Conflict**: an inconsistent subset of U: show one cause of inconsistency.
- **Relaxation**: a consistent subset of U: show one possible way of recovering from it

Optimality – sort of

- A relaxation is **maximal** when *no constraint can added* while remaining consistent.
- A conflict is **minimal** when *no constraint can be removed* while remaining inconsistent.

Example

Car configuration

Option	Cost
Roof rack	500
Convertible	500
CD Player	500
Leather Seats	2600

➠ Convertible cars cannot have roof racks.

User constraints

c_1	Total cost ≤ 3000
c_2	Roof rack
c_3	Convertible
c_4	CD Player
c_5	Leather Seats

Relaxations: $\{c_1 c_2\}$, $\{c_1 c_5\}$ are consistent

Maximality: $\{c_1 c_2 c_4\}$ is still consistent, but no more constraint can be added to $\{c_1 c_5\}$.

Explanation by Proof

Example Problem

- $y = 5x_1 + 4x_2 + 26x_3$
- $x_1 = 1$
- $x2 = 1$
- $x3 = 1$
- $y \leq 30$

Proof by Propagation

Order the constraints lexicographically:

- $A \wedge B \implies y \geq 5$
- $A \wedge C \wedge y \geq 5 \implies y \geq 9$
- $A \wedge D \wedge y \geq 9 \implies y \geq 35$
- $A \wedge E \wedge y \geq 35 \implies \perp$

Explanation: $\{A, B, C, D, E\}$, which is not minimal.

From Proofs to Checks

Proofs

- Find an inconsistency proof with minimal explanation
- **Non-Decomposable:** a proof with non-minimal explanation need not contain a proof with minimal explanation

Checks

- Find a minimal inconsistent subset of the constraints
- **Decomposable:** a non-minimal explanation contains a minimal explanation

The QuickXplain Algorithm

- **Explanation subproblem**: background B and constraints C
- **Task**: find minimal subset X of C s.t. B and X fail
- **Method**: initial problems is split into subproblems constraints may be moved from C to B constraints may be omitted from C

QuickXplain's Principles

- **Skip rule**: if B is inconsistent then $X = \{\}$
- **Culprit rule**: if B is consistent and $C = \{c\}$ then $X = \{c\}$
- **Decomposition rule**: if B is consistent and $C = C_1 \cup C_2$
  - find explanation X_2 of subproblem $B \cup C_1, C_2$ and
  - find explanation X_1 of subproblem $B \cup X_2, C_1$
  - result X is the union of X_1 and X_2

How to use QuickXplain

- **Background:** effort is reduced by putting as many constraints as possible in the initial background
- **Preference order**: order of constraint uniquely characterizes the conflict found
- **Consistency checker**: time can be traded against minimality by an incomplete consistency checker, giving “anytime” behaviour

Applications of QuickXplain

- Configuration: B2B, B2C find conflicts between user requests.
- Constraint model debugging isolate failing parts of the constraint model.
- Rule verification find tests that make a rule never applicable.
- Benders decomposition.
- Diagnosis of ontologies.

Proofs versus Checks

Truth Maintenance

- proof-based (syntactic)
- computed online
- tight interaction with problem solver
- explanation needs not be minimal
- high space complexity

Consistency-based

- consistency check-based (semantic)
- computed off-line
- uses problem solver as black-box
- minimal (preferred) explanation
- high time complexity

Introduction

- Presenting a representative set of explanations
- Joint work with Alexandre Papadopulous (4C), Boi Faltings and Pearl Pu (EPFL)
- Published at AAAI 2007
- Forthcoming related paper at CP
- Funded by Science Foundation Ireland

Observations

1. **Conflict**: doesn't guide the user to solving the problem
2. **Single relaxation**: may not satisfy the user desires
3. **All relaxations**: can theoretically be too large

An Alternative Approach

- show a set of relaxations
- that must be *representative* of all possible relaxations

as a trade-off between compactness and comprehensiveness

Explanations

- **Relaxation** a consistent subset of U (what we keep)
- **Exclusion set** the complement of a relaxation (what we exclude)
- **Explanation** a relaxation together with its corresponding exclusion set
- **Conflict** an inconsistent subset of U

Optimality

- A relaxation is *Maximal* when no constraint can added while remaining consistent
- A conflict is *Minimal* when no constraint can be removed while remaining inconsistent

Example

Car configuration

Option	Cost
Roof rack	500
Convertible	500
CD Player	500
Leather Seats	2600

➠ Convertible cars cannot have roof racks.

User constraints

c_1	Total cost ≤ 3000
c_2	Roof rack
c_3	Convertible
c_4	CD Player
c_5	Leather Seats

Relaxations: $\{c_1 c_2\}$, $\{c_1 c_5\}$ are consistent
Maximality: $\{c_1 c_2 c_4\}$ is still consistent, but no more constraint can be added to $\{c_1 c_5\}$.

Definition

Representative set of explanations

c_1	c_2	c_3	c_4	c_5
✗	✗	✓	✓	✓
✗	✓	✗	✓	✓
✓	✗	✓	✓	✗
✓	✓	✗	✓	✗
✓	✗	✗	✗	✓

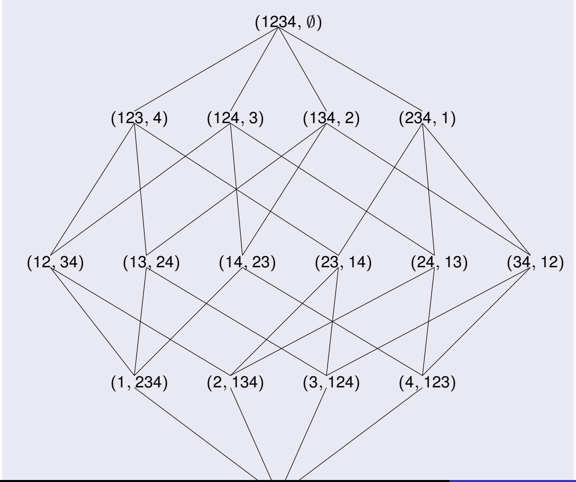
- Every constraint that can be kept is kept at least once
- Every constraint that can be relaxed is relaxed at least once
- Minimal (setwise) representative set of explanations

Complexity

Decision problems

- Does a maximal relaxation contain a given constraint?
  ➥ Polynomial (in terms of number of calls to the consistency checker)
- Does a minimal exclusion set contain a given constraint?
  ➥ *NP*-Complete (with an oracle for the consistency checker)

Finding All Relaxations (D&A Bailey et. al 2005)



Problem

Explanations:
$(12, 34)$
$(13, 24)$
$(4, 123)$
Conflicts:
$14, 23, 24, 34$

Representative Explanations

Goal

Speed up the convergence of the complete method to a representative set of explanations

Two points of choice

1. Which new entry point to choose?
2. Which parent to choose?

Heuristics

1. Choose a consistent set that becomes a conflict with an uncovered constraint
2. Add covered constraints first

Empirical Analysis

Random problems

- 15 variables,
- One background table constraint, with varying tightness
- Random assignments on the variables

Renault

- Real-world problem
- 99 variables
- 2.8×10^{12} solutions
- 30 variables randomly assigned

Behaviour

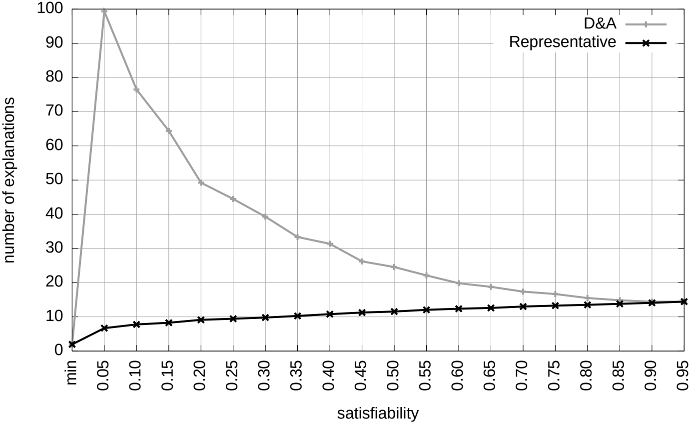


Figure: Number of explanations

Behaviour

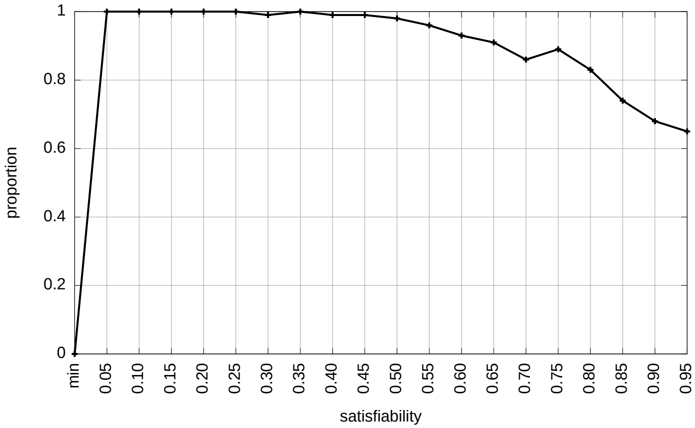
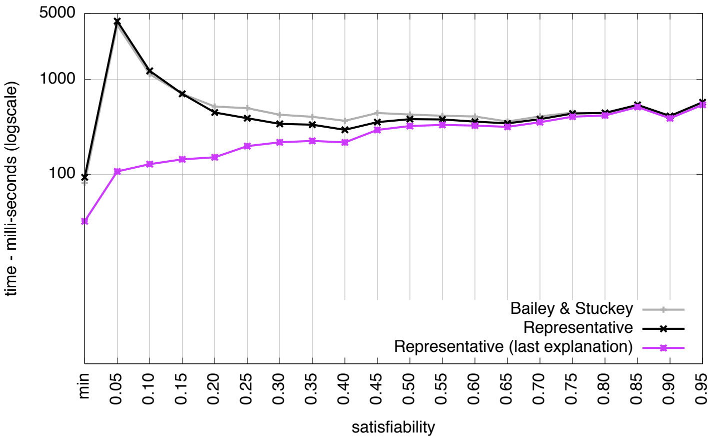


Figure: Proportion of “true instances”

Empirical Analysis



Empirical Analysis

Renault instance

Instance	Baseline		REPRESENTATIVEXPLAIN		
	time	#exps	time last	time all	#exps
renault 10^6	474.76	17	318.87	618.76	3
renault 10^7	263.95	11	125.51	324.71	3
renault 10^8	205.82	8	97.98	232.32	3
renault 10^9	293.00	12	139.67	350.51	3

Table: Running times for the Renault instances

Introduction

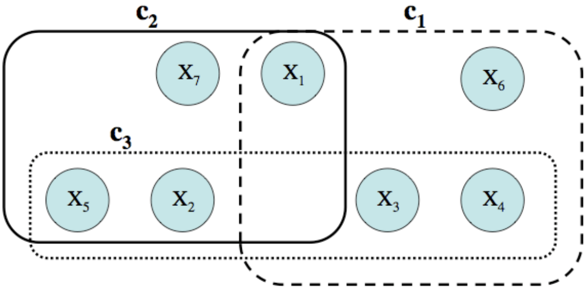
- Automatically reformulating constraints for explanation
- Joint work with Hadrien Cambazard
- Google Best Paper at AICS 2007, published in Constraints
- Forthcoming paper at CP
- Funded by Science Foundation Ireland

Motivation

- Many problems involve large arity table constraints
- Spreadsheets, databases, catalogues, etc.
- Algorithms such as QuickXplain are constraint-centred
- Large arity constraints can “hide” the conflict, and result in useless explanations

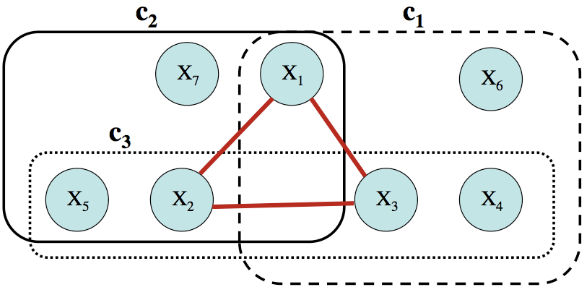
Example

The following problem is defined in terms of three 4-ary constraints



Example

...but it might be possible to reformulate to focus on binary conflicts



Functional Dependencies

We exploit functional dependencies, usually used to normalize database tables, to reformulate constraints

x1	x2	x3	x4
0	0	0	4
0	4	2	4
1	0	0	2
2	2	3	2
2	4	1	3

x3 --> x2 : x3 determines x2

~~x2 --> x3~~

{x1,x2} --> x4

Reformulation Approach

The basic procedure is as follows:

x1	x2	x3	x4
0	0	0	4
0	4	2	4
1	0	0	2
2	2	3	2
2	4	1	3

{x1,x2} --> x4

x1	x2	x4
0	0	4
0	4	4
1	0	2
2	2	2
2	4	3

x1	x2	x3
0	0	0
0	4	2
1	0	0
2	2	3
2	4	1

x3 --> x2

x1	x2	x4
0	0	4
0	2	4
1	0	2
2	2	2
2	4	3

x1	x2
0	0
1	0
2	1
0	2
2	3

x2	x3
0	0
4	1
4	2
2	3

Normal forms in data base : 3NF, BCNF

Minimal decomposition

Properties of the Reformulation

The reformulation we obtain has some nice properties:

Lossless: the set of solutions is preserved

Propagation: pruning is equivalent in the reformulation as the original

What is a Good Reformulation?

Two criteria:

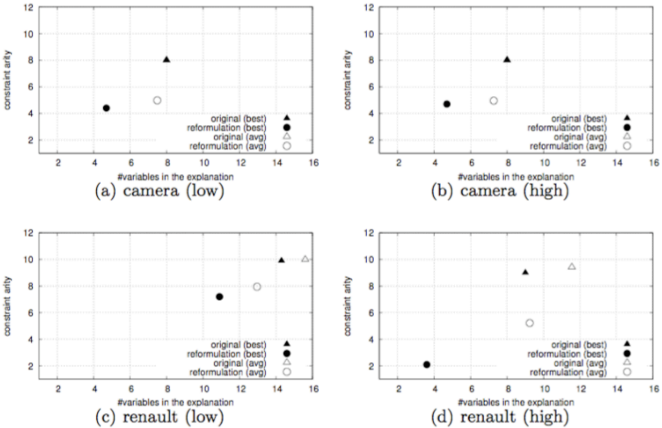
Arity: for explanations we want to minimize the maximum arity of the constraints in the reformulation

Memory Size: we might want to minimize the memory footprint of the reformulation

Functional Dependencies in Real Constraints

Data-set	#tuples	arity	#dependencies	#constraints	min.arity	max.arity	time(s)
camera	113	8	41	4	5	5	0.20
laptop	403	10	54	4	5	6	0.57
renault R80	342	10	2	3	2	8	0.00
renault R104	164	9	11	6	2	4	0.02
travel R0	1470	9	7	4	4	6	0.00

Improved Explanations

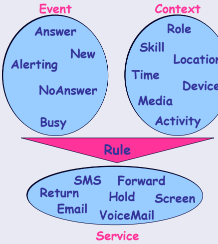


Introduction

- Personalized Telecom Feature Subscription
- This is joint work between:
  - David Lesaint (BT)
  - Deepak Mehta, Barry O'Sullivan, Luis Quesada and Nic Wilson (4C)
  - Funded by IRCSET Enterprise Partnership
  - Forthcoming papers in 2008: AAAI/IAAI, ECAI/PAIS, and CP.

Introduction

Context-Aware Personalized Services



User intentions

- If I am in a meeting, divert calls to my mobile
- If I am out of the office, play an announcement and text me on my mobile.
- bar international calls at off-peak time in my department.

Call Control Features

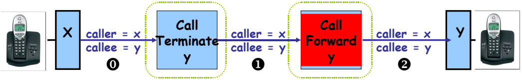
- Call control features are the building blocks to achieve personalization.
- Each feature can be seen as an increment of the basic functionality.

Catalogue of Features

Feature	**Acronym**
Call Forwarding Unconditional	**CFU**
Call Terminate	**CT**
Number Translation	**NT**
Find Me	**FM**
Call Forwarding on Busy	**CFB**
Originating Call Logging	**OCL**
Originating Call Screening	**OCS**

Feature Interactions

- Features may modify or influence one another.
- There are *desirable interactions* as well as *undesirable interactions*.



- The creation of a personalized service is subject to **integrity constraints**.
- The integrity constraints are precedence relations that avoid undesirable behaviors.

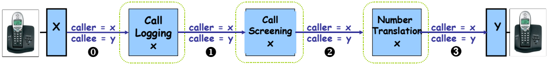
User Preferences

- When a set of features are put together by a user many different behavioral options for his personalized service might exist.
- A user will need to choose among these options by providing his/her **preferences** in terms of precedence relations between features.

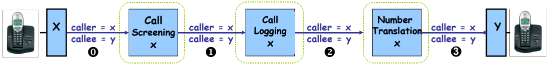
User Preferences: Example

Let us suppose that a person subscribes to three features: Originating call screening, Number translation and Originating call logging.

- screen on the dialed number and log every call attempt

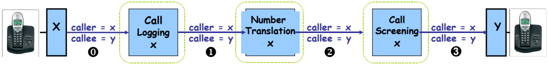


- screen on the dialed number and log only the successful call

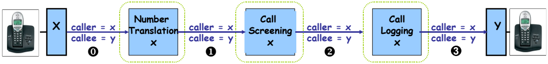


User Preferences: Example

- screen on the translated number and log every call attempt



- screen on the translated number and log only the successful call



Consistent Feature Subscription

- A subscription is consistent if it satisfies the integrity constraint and the precedence relations.
- The goal is to find a consistent subscription that is optimal with respect to user defined precedence relations.
- If no consistent subscription can be found, then the goal is to find the best relaxation of the feature subscription.
- Finding the best relaxation of an inconsistent subscription is NP-Hard.

Approaches

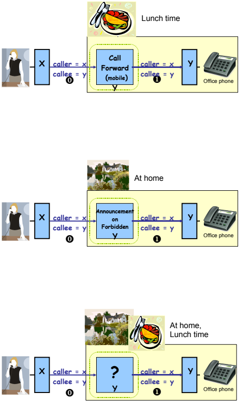
- We model and solve our feature subscription configuration problem using three different approaches:
  - Constraint Programming
  - Satisfiability (SAT)
  - Integer Linear Programming (ILP).
- An important advantage of CP is its expressiveness for capturing the constraints arising in this telecommunication domain.
- Non trivial improvements to the CP model are required for it to be competitive with the SAT approach we used.

Context-dependent Consistent Feature Subscription

Context-aware telecommunication services involves researching and developing methods for providing **context-dependent consistent feature subscriptions** to end-users by resolving various issues such as:

- feature interaction management,
- representation and handling of context information from a service configuration perspective,
- identifying the main context dimensions,
- conflict-resolution mechanisms, and
- the management of priorities and preferences.

An example of context-dependent conflict



Feature Subscription: Input

- A source subscription set S. It is a subset of the source features defined in the catalog.
- A target subscription set T. It is a subset of the target features defined in the catalog.
- A set of constraints C_S defined over the set S in the catalog, e.g. precedence constraint, incompatibility constraint etc.
- A set of constraints C_T defined over the target subscription set T.
- A set of user preferences P_S defined over the source subscription set S.
- A set of user preferences P_T defined over the target subscription set T.

Feature Subscription: Zones

- A source zone is a sequence of features available in the source subscription set.
- A target zone is a sequence of features available in the target subscription set.

Consistent Feature Subscription

A feature subscription is **consistent** iff it satisfies the following:

- The source zone is consistent with the constraints defined in the set C_S.
- The target zone is consistent with the constraints defined in the set C_T.
- For every reversible feature f, f is a part of the source zone iff f is a part of the target zone.
- For every reversible feature $f1$ and $f2$, $f1$ precedes $f2$ in the source zone iff $f2$ precedes $f1$ in the target zone.

What is the problem?

- The problem is to find a **consistent feature subscription** that is optimal with respect to user defined preferences.
- If the feature subscription is inconsistent then we have to find a **maximally preferred relaxation** of the feature subscription.
- We can relax the problem by dropping features and/or by discarding user precedence relations.

An example of a feature subscription (I)

Catalog

source catalogue

	F1	F2	F5	F6	F7
F1		<	<	<	<
F2	>		<	<	<
F5	>	>		<	<
F6	>	>	>		<
F7	>	>	>	>	

target catalogue

	F3	F4	*F5*	F6	F7	F8
F3		<>	>	>	>	>
F4	<>		>	>	>	
F5	<	<		>	>	>
F6	<	<	<		>	>
F7	<	<	<	<		>
F8	<		<	<	<	

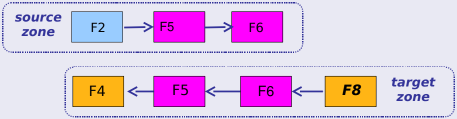
Feature Subscription Input

- $S = \{F2, F5, F6\}$
- $C_S = \{F2 < F5, F2 < F6, F5 < F6\}$
- $P_S = \{\}$
- $T = \{F4, F5, F6, F8\}$
- $C_T = \{F4 > F5, F4 > F6, F5 > F6, F5 > F8, F6 > F8\}$
- $P_T = \{F4 > F8\}$

An example of a feature subscription (II)

- $S = \{F2, F5, F6\}$
- $C_S = \{F2 < F5, F2 < F6, F5 < F6\}$
- $P_S = \{\}$
- $T = \{F4, F5, F6, F8\}$
- $C_T = \{F4 > F5, F4 > F6, F5 > F6,$
  $F5 > F8, F6 > F8\}$
- $P_T = \{F4 > F8\}$

Consistent Subscription



Composition of source and target catalog (I)

source catalogue

	F1	F2	F5	F6	F7
F1		<	<	<	<
F2	>		<	<	<
F5	>	>		<	<
F6	>	>	>		<
F7	>	>	>	>	

target catalogue

	F3	F4	F5	F6	F7	F8
F3		<>	>	>	>	>
F4	<>		>	>	>	
F5	<	<		>	>	>
F6	<	<	<		>	>
F7	<	<	<	<		>
F8	<		<	<	<	

Composition of source and target catalogue

	F1	F2	F3	F4	F5	F6	F7	F8
F1		<			<	<	<	
F2	>				<	<	<	
F3				<>	<	<	<	<
F4			<>		<	<	<	
F5	>	>	>	>		<	<	<
F6	>	>	>	>	>		<	<
F7	>	>	>	>	>	>		<
F8			>		>	>	>	

Source-only
Reversible
Target-only

Composition of source and target catalog (II)

catalogue

	F1	F2	F3	F4	F5	F6	F7	F8
F1		<			<	<	<	
F2	>				<	<	<	
F3				<>	<	<	<	<
F4			<>		<	<	<	
F5	>	>	>	>		<	<	<
F6	>	>	>	>	>		<	<
F7	>	>	>	>	>	>		<
F8			>		>	>	>	

User Subscription: Input

F = {F2, F4, F5, F6, F8}

C = {F2 < F5, F2 < F6, F5 < F6,
F4 < F5, F4 < F6, F5 < F6,
F5 < F8, F6 < F8}

P = {F4 < F8}

F2 → F4 → F5 → F6 → F8

Composition of source and target catalog (III)

catalogue

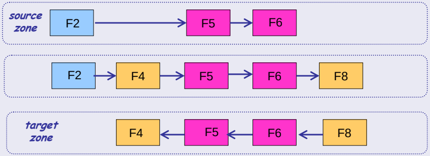
	F1	F2	F3	F4	F5	F6	F7	F8
F1		<			<	<	<	
F2	>				<	<	<	
F3				<>	<	<	<	<
F4			<>		<	<	<	
F5	>	>	>	>		<	<	<
F6	>	>	>	>	>		<	<
F7	>	>	>	>	>	>		<
F8			>		>	>	>	

User Subscription: Input

F = {F2, F4, F5, F6, F8}

C = {F2 < F5, F2 < F6, F5 < F6,
F4 < F5, F4 < F6, F5 < F6,
F5 < F8, F6 < F8}

P = {F4 < F8}



Feature Subscription

- The input of the feature subscription problem can now be simplified as follows:
  - a set of features F from the catalog,
  - a set of constraints C defined over F from the catalog, and
  - a set of user precedence relations P.
- A feature subscription is consistent iff a total order can be established on the features in F by satisfying the constraints in C.
- The problem is to find a consistent subscription that is optimal with respect to user defined precedences.
- If no consistent subscription can be found, then the problem is to find the best relaxation of the feature subscription, which is consistent.

Computing Minimal Conflicts

- Constraints are added step by step until a failure is detected.
- The last constraint added participates in at least one minimal conflict.
- Example: $\{c_1, c_2, c_3, c_4, c_5\}$ is not satisfiable because c_1 is not compatible with c_5.

step	activated constraint	result	partial conflict
1	c_1	no fail	$\{\}$
2	$c_1\ c_2$	no fail	$\{\}$
3	$c_1\ c_2\ c_3$	no fail	$\{\}$
4	$c_1\ c_2\ c_3\ c_4$	no fail	$\{\}$
5	$c_1\ c_2\ c_3\ c_4\ c_5$	fail	$\{c_5\}$
6	c_5	no fail	$\{c_1\}$
7	c_1	fail	$\{c_1, c_5\}$

Approaches for Computing Minimal Conflicts (I)

- QuickXplain
  - Computes only one conflict.
  - Follows a Divide and Conquer approach.
  - Time complexity: $\mathcal{O}(n\,log(k+1))$.
- De la Banda et al's, or Bailey and Stuckey's approaches.
  - Computes all the minimal conflicts.
  - Explores the subsets of the given set of constraints in a smart way.
  - Avoids subsets of satisfiable sets and supersets of conflicts.

Approaches for Computing Minimal Conflicts (II)

- Dualize and Advance
  - Computes both all the minimal conflicts and all maximal relaxations.
  - Relies on the notion of hitting sets.
  - Both time and space complexities are exponential in term of the number of constraints.
  - It is not suitable when the number of conflicts is too high.
- Backtrack Search
  - Computes all the minimal conflicts.
  - Avoids subsets of satisfiable sets and supersets of conflicts.
  - Time complexity is exponential but space complexity is linear.
  - It is suitable when the number of conflicts is too high.

Branch and Bound (I)

- Branch and Bound is a general algorithmic method for finding optimal solutions.
- We use it to find the best relaxation of an inconsistent feature subscription.
- It is basically an enumeration approach in a fashion that prunes the non-promising search space.

Modeling the problem as a COP(I)

- **Variables and Domains.**
  - We associate each feature $f_i \in F$ (the set of selected features) with two variables:
    - bf_i is a Boolean variable. It is set to 1 or 0 depending on whether f_i is included in the consistent subscription or not.
    - pf_i is a position variable. It represents the position of f_i. The domain of pf_i is the set of available positions.
  - We associate a Boolean variable bp_{ij} with each user precedence relation $p_{ij} \equiv f_i < f_j$ in P (the set of user precedence relations).

Modeling the problem as a COP(II)

- **Constraints**
  - Precedence constraints in catalog

$$bf_i \wedge bf_j \rightarrow (pf_i < pf_j)$$

  - Precedence constraints defined by the user (Preference)

$$bp_{ij} \rightarrow (bf_i \wedge bf_j \wedge (pf_i < pf_j))$$

  - Incompatibility constraints in catalog

$$bf_i \neq bf_j$$

Modeling the problem as a COP(III)

- **Objective Function**
  The objective is to maximize:

$$\sum_{f_i \in F} bf_i \times wf_i + \sum_{p_{ij} \in P} bp_{ij} \times wp_{ij}$$

Partial Weighted MaxSAT (I)

- Boolean Satisfiability Problem (SAT) is a decision problem whose instance is a Boolean expression written using only $\wedge$, $\vee$, $\neg$, variables and parenthesis.
- The problem is to decide whether there is an assignment of true and false values to the variables that will make the expression true.
- The expression is normally written in Conjunctive Normal Form like $(p \vee q \vee r) \wedge (q \vee w \vee s) \wedge ...(r \vee t \vee q)$.
- Partial Weighted MaxSAT is an extension of SAT which includes the notions of hard and soft clauses.
- The idea is to find an assignment that maximizes the cost.

Partial Weighted MaxSAT (II)

The Feature Subscription Problem can be modeled as a Partial Weighted MaxSAT problem as follows:

- Precedence constraints in the catalog:

$$\frac{p_{ij} \in C}{\langle \top, (\neg bf_i \vee \neg bf_j \vee bp_{ij}) \rangle \in SatInst}$$

- The precedence relation is transitive:

$$\frac{\{p_{ij}, p_{jk}\} \subseteq C \cup P}{\langle \top, (\neg bp_{ij} \vee \neg bp_{jk} \vee bp_{ik}) \rangle \in SatInst}$$

- The precedence relation is antisymmetric:

$$\frac{p_{ij} \in C \cup P}{\langle \top, (bp_{ij} \vee bp_{ji}) \rangle \in SatInst} \quad \frac{p_{ij} \in C \cup P}{\langle \top, (\neg bp_{ij} \vee \neg bp_{ji}) \rangle \in SatInst}$$

Partial Weighted MaxSAT (III)

- Each feature is associated with its weight:

$$\frac{\langle wf_i, f_i\rangle \in F}{\langle wf_i, (bf_i)\rangle \in SatInst}$$

- Each user precedence relation is associated with its weight:

$$\frac{\langle wp_{ij}, p_{ij}\rangle \in P}{\langle wp_{ij}, (bp_{ij})\rangle \in SatInst}$$

- A user precedence relation is only satisfied if its features are included:

$$\frac{\langle wp_{ij}, p_{ij}\rangle \in P}{\langle \top, (bf_i \vee \neg bp_{ij})\rangle \in SatInst} \quad \frac{\langle wp_{ij}, p_{ij}\rangle \in P}{\langle \top, (bf_j \vee \neg bp_{ij})\rangle \in SatInst}$$

Integer Linear Programming (I)

- Maximize

$$\sum_{f_i \in F} wf_i bf_i + \sum_{p_{ij} \in P} wp_{ij} bp_{ij}$$

- Catalog Precedence Constraint

$$bf_i + bf_j - C_{ij} \leq 1$$

$$pf_i - pf_j + n * C_{ij} \geq 1$$

$$pf_i - pf_j + n * C_{ij} \leq n - 1$$

- Catalog incompatibility constraint

$$bf_i + bf_j \leq 1$$

Integer Linear Programming (II)

- User Precedence Preference

$$bf_i - P_{ij} \geq 0$$

$$bf_j - P_{ij} \geq 0$$

$$pf_i - pf_j + n * P_{ij} \geq 1$$

$$pf_i - pf_j + n * P_{ij} \leq n - 1$$

Some Empirical Results

Random catalogue = $\langle 50, 250, \{<, >\}\rangle$

User Subscription		SAT	CPLEX	CP	CP+
$\langle 10, 5, 4, \{<\}\rangle$	#Nodes	26	0	11	13
(value=27)	Time	0.78	0.06	0.04	0.06
$\langle 20, 10, 4, \{<\}\rangle$	#Nodes	670	1	65	38
(value=57)	Time	3.01	0.06	0.18	0.12
$\langle 30, 20, 4, \{<\}\rangle$	#Nodes	1,848	29,668	66,835	11,629
(value=85)	Time	9.36	59.61	18.92	2.65
$\langle 40, 40, 4, \{<\}\rangle$	#Nodes	47,502	1,565,793	50,274,725	1,091,194
(value=117)	Time	66.01	9,101.01	18,562.25	409.86

Explanations in Constraint Programming

Barry O'Sullivan

Cork Constraint Computation Centre
Department of Computer Science
University College of Cork, Ireland

email: `b.osullivan@cs.ucc.ie`

ACP Summer School 2008