PackUp: Tools for Package Upgradability Solving
SYSTEM DESCRIPTION

Mikoláš Janota[1] Inês Lynce[1]
Vasco Manquinho[1] Joao Marques-Silva[1,2]

[1] INESC-ID/IST, Lisbon, Portugal
[2] CASL/CSI, University College Dublin, Ireland
supported by BEACON, MANCOOSI, BSOLO, and iExplain

Package Management Systems

```
install p
```

```
remove p
```

Package Management Systems

```
install p
```

```
remove p
```

- may install other packages on which it depends
- may uninstall other packages with which it conflicts

Why is it Hard?

- `p` may depend on `n` or `q`

Why is it Hard?

- `p` may depend on `n` or `q`

```
install p
p depends n OR q
n conflicts z
```

Why is it Hard?

- `p` may depend on `n` or `q`

```
install p
p depends n OR q
n conflicts z
```

$$\begin{array}{lcl} x_p & & \\ x_p & \Longrightarrow & x_n \vee x_q \\ x_n & \Longrightarrow & \neg x_z \\ \hline \end{array}$$

Why is it Hard?

- `p` may depend on `n` or `q`

```
install p
p depends n OR q
n conflicts z
```

$$\frac{\begin{array}{lcl} x_p & & \\ x_p & \Longrightarrow & x_n \vee x_q \\ x_n & \Longrightarrow & \neg x_z \end{array}}{\neg x_z \vee x_q}$$

Why is it Hard?

- `p` may depend on `n` or `q`

```
install p
p depends n OR q
n conflicts z
```

$$\frac{\begin{array}{rcl} x_p & & \\ x_p & \implies & x_n \vee x_q \\ x_n & \implies & \neg x_z \end{array}}{\neg x_z \vee x_q}$$

- Deciding whether a package can be installed or not is NP-complete.

Is Any Solution Good?

Scenario	Solutions

- `p depends on q OR r`
- `q.installed=false`
- `r.installed=false`

Is Any Solution Good?

Scenario

- `p depends on q OR r`
- `q.installed=false`
- `r.installed=false`

Solutions

1. install one of `q`, `r`
2. install both `q`, `r`

Is Any Solution Good?

Scenario

- `p depends on q OR r`
- `q.installed=false`
- `r.installed=false`

- `p depends on q OR r`
- `q conflicts with n`
- `n.installed=true`

Solutions

1. install one of `q`, `r`
2. install both `q`, `r`

Is Any Solution Good?

Scenario | **Solutions**

- `p depends on q OR r`
- `q.installed=false`
- `r.installed=false`

1. install one of `q`, `r`
2. install both `q`, `r`

- `p depends on q OR r`
- `q conflicts with n`
- `n.installed=true`

1. install `y` and remove `q`
2. install `z` and keep `q`

Is Any Solution Good?

Scenario	Solutions
• `p depends on q OR r` • `q.installed=false` • `r.installed=false`	1. install one of `q`, `r` 2. install both `q`, `r`
• `p depends on q OR r` • `q conflicts with n` • `n.installed=true`	1. install `y` and remove `q` 2. install `z` and keep `q`

Morale

- some configurations are more preferable than others

Problem Definition

- Each package has ...
  - a name and a version.
  - a set of conflicting packages.
  - dependencies (CNF).
  - information whether the package's currently installed or not.

Problem Definition

- Each package has ...
  - a name and a version.
  - a set of conflicting packages.
  - dependencies (CNF).
  - information whether the package's currently installed or not.
- Preference over solutions given by a lexicographic criterion $(f_1, \ldots, f_n)$

Problem Definition

- Each package has ...
  - a name and a version.
  - a set of conflicting packages.
  - dependencies (CNF).
  - information whether the package's currently installed or not.
- Preference over solutions given by a lexicographic criterion $(f_1, \ldots, f_n)$

Example

```
package:  p
version:  1
depends:  q>=5, r=3
conflicts:  x!=1, n
```

```
-new,-removed
```

Partial Weighted MaxSAT

Problem

- a set of hard clauses
- a set of soft clauses, (W, c)

Partial Weighted MaxSAT

Problem

- a set of hard clauses
- a set of soft clauses, (W, c)

Solution

An variable valuation that

- satisfies all hard clauses
- maximizes the sum of weights satisfied soft clauses

Partial Weighted MaxSAT

Problem

- a set of hard clauses
- a set of soft clauses, (W, c)

Solution

An variable valuation that

- satisfies all hard clauses
- maximizes the sum of weights satisfied soft clauses

- Can be easily translated to OPB and other similar formalisms.

Problem to MaxSAT

- Hard clauses represent conflicts and dependencies.
- Soft clauses represent preference.

Problem to MaxSAT

- Hard clauses represent conflicts and dependencies.
- Soft clauses represent preference.
- Weights chosen to represent the lexicographic ordering, i.e., for a criterion $(f_1, \ldots, f_n)$

$$W_i = 1 + \Sigma_{i<j} W_j \times c_j$$

where c_j is the number of clauses generated for the function f_j.

Encoding (straightforward version)

- for each package (p, v) introduce a variable x_p^v

Encoding (straightforward version)

- for each package (p, v) introduce a variable x_p^v
- `(p,1).depends = z>=3` as $\neg x_p^1 \vee x_z^3 \vee \cdots \vee x_z^k$

Encoding (straightforward version)

- for each package (p, v) introduce a variable x_p^v
- `(p,1).depends = z>=3` as $\neg x_p^1 \vee x_z^3 \vee \cdots \vee x_z^k$
- `(p,1).conflicts = q=3` as $\neg x_p^1 \vee \neg x_q^3$
- `install p` as $x_p^1 \vee \cdots \vee x_p^l$

Encoding (straightforward version)

- for each package (p, v) introduce a variable x_p^v
- `(p,1).depends = z>=3` as $\neg x_p^1 \vee x_z^3 \vee \cdots \vee x_z^k$
- `(p,1).conflicts = q=3` as $\neg x_p^1 \vee \neg x_q^3$
- `install p` as $x_p^1 \vee \cdots \vee x_p^l$

- preferences in an analogous fashion, e.g. `p=3` should stay installed

$$(W, x_p^3)$$

Encoding Versions

Common Intervals

`(p,1).depends = z>=3` as $\neg x_p^1 \vee x_z^3 \vee x_z^4 \vee \ \ x_z^5 \cdots \vee x_z^k$

`(q,1).depends = z>=5` as $\neg x_q^1 \vee x_z^5 \vee \cdots \vee x_z^k$

Encoding Versions

Common Intervals

`(p,1).depends = z>=3` as $\neg x_p^1 \vee x_z^3 \vee x_z^4 \vee$ $x_z^5 \cdots \vee x_z^k$

`(q,1).depends = z>=5` as $\neg x_q^1 \vee$ $x_z^5 \vee \cdots \vee x_z^k$

Encoding Versions

Common Intervals

`(p,1).depends = z>=3` as $\neg x_p^1 \vee x_z^3 \vee x_z^4 \vee$ $x_z^5 \cdots \vee x_z^k$

`(q,1).depends = z>=5` as $\neg x_q^1 \vee$ $x_z^5 \vee \cdots \vee x_z^k$

Interval Joining

$$\neg x_q^1 \vee x_z^3 \vee x_z^4 \vee \mathrm{i}{\uparrow}_z^5$$
$$\neg x_q^1 \vee \mathrm{i}{\uparrow}_z^5$$

Interval variables

Introduce Fresh Variables Representing Intervals

- $\mathrm{i}{\uparrow}_p^v$ — a version greater than or equal to v of p is installed
- $\mathrm{i}{\downarrow}_p^v$ — a version less than or equal to v of p is installed
- $\mathrm{u}{\uparrow}_p^v$ — versions greater than or equal to v of p are uninstalled
- $\mathrm{u}{\downarrow}_p^v$ — versions less than or equal to v of p are uninstalled

Interval variables

Introduce Fresh Variables Representing Intervals

- $\mathrm{i}{\upharpoonleft}_p^v$ — a version greater than or equal to v of p is installed
- $\mathrm{i}{\downharpoonleft}_p^v$ — a version less than or equal to v of p is installed
- $\mathrm{u}{\upharpoonleft}_p^v$ — versions greater than or equal to v of p are uninstalled
- $\mathrm{u}{\downharpoonleft}_p^v$ — versions less than or equal to v of p are uninstalled

Interval Variables' Semantics Is Defined Inductively

$$\neg\mathrm{i}{\upharpoonleft}_p^v \vee x_p^v \vee \mathrm{i}{\upharpoonleft}_p^{v+1}$$
$$(\neg\,\mathrm{u}{\upharpoonleft}_p^v \vee \neg x_p^v) \wedge (\neg\,\mathrm{u}{\upharpoonleft}_p^v \vee \mathrm{u}{\upharpoonleft}_p^{v-1})$$
$$\ldots$$

Computing Lexicographic Optimization

- Solvers tend to work poorly for large weights.
- Solve iteratively, i.e. minimize each function separately.

Computing Lexicographic Optimization

- Solvers tend to work poorly for large weights.
- Solve iteratively, i.e. minimize each function separately.

minimizing criterion $(f_1, \ldots, f_n)$

```
1 for $i \leftarrow 1 \ldots n$ do
2     $v_i \leftarrow$ minimize($f_i$) in $\phi$
3     $\phi \leftarrow \phi \wedge (f_i = v_i)$
```

Invoking PackUP

- MaxSAT solver—solver invoked just once

```
--max-sat \
--external-solver 'msuncore -wl -bmo'
```

Invoking PackUP

- MaxSAT solver—solver invoked just once

```
--max-sat \
--external-solver 'msuncore -wl -bmo'
```

- OPB solver—solver invoked multiple times

```
--external-solver 'minisatp' \
--multiplication-string ' '
```

Summary

- PackUP enables solving package upgradability problem with an external solver
- Instantiations `cudf2msu` and `cudf2opb` participated in the 3rd MISC Live, winning 4/5 tracks.
- The solver can be a MaxSAT or OPB.
- The use of OPB enables iterative approach to lexicographic optimization
- Package versions are encoded using interval variables.
- Released under GPL
  `http://sat.inesc-id.pt/~mikolas/sw/packup`