CVC4 - the SMT Solver

Installation on Linux

- #install make, for example: apt-get install build-essential
- #install libgmp, for example: apt-get install libgmp-dev
- #install boost, for example: apt-get install libboost-all-dev
- wget http://cvc4.cs.nyu.edu/builds/src/cvc4-1.4.tar.gz
- tar -zxvf cvc4-1.4.tar.gz
- cd cvc4-1.4/contrib
- ./get-antlr-3.4
- cd ..
- ./configure --with-antlr-dir=`pwd`/antlr-3.4 ANTLR=`pwd`/antlr-3.4/bin/antlr3
- make

Installation on Windows

- Binary: http://cvc4.cs.nyu.edu/builds/win32-opt/cvc4-1.4-win32-opt.exe
- Rename to: cvc4.exe
- Move to: C:\cvc4\cvc4.exe
- Append environment variable $PATH: C:\cvc4

Installation on Mac OS

- Mountain Lion: http://cvc4.cs.nyu.edu/builds/macos/cvc4-1.4.MacOs85.MountainLion.mpkg
- Mavaricks +: http://cvc4.cs.nyu.edu/builds/macos/cvc4-1.4_0.MacOs9.Mavericks.mpkg

Example

- Execute cvc4 interactive mode:

```
cvc4 –-lang smt
```

- Declares a value p and asks whether (p∧¬p) is satisfiable:

```
>(set-logic QF_UF)
>(declare-fun p () Bool)
>(assert (and p (not p)))
>(check-sat)
unsat
>(exit)
```

Set Option

```
> (set-option :print-success true)
> (set-logic QF_UF)
success
> (declare-fun p () Bool)
success
> (assert (and p (not p)))
success
> (check-sat)
unsat
> (exit)
```

Integer Arithmetic

- x + 2*y = 20 and x - y = 2 ?

```
> (set-logic QF_LIA)
success
> (declare-fun x () Int)
success
> (declare-fun y () Int)
success
> (assert (= (+ x (* 2 y)) 20))
success
> (assert (= (- x y) 2))
success
> (check-sat)
sat
> (exit)
success
```

Integer Arithmetic

- x + 2*y = 20 and x - y = 2 ?

```
> (set-logic QF_LIA)
success
> (declare-fun x () Int)
success
> (declare-fun y () Int)
success
> (assert (= (+ x (* 2 y)) 20))
success
> (assert (= (- x y) 3))
success
> (check-sat)
unsat
> (exit)
success
```

Real Arithmetic

- x + 2*y = 20 and x - y = 2 ?

```
> (set-logic AUFNIRA)
success
> (declare-fun x () Real)
success
> (declare-fun y () Real)
success
> (assert (= (+ x (* 2 y)) 20))
success
> (assert (= (- x y) 3))
success
> (check-sat)
sat
> (exit)
success
```

To Real

```
>(set-logic AUFNIRA)
>(declare-const a Int)
>(declare-const b Int)
>(declare-const c Int)
>(declare-const d Real)
>(declare-const e Real)
>(assert (> e (+ (to_real (+ a b)) 2.0)))
>(assert (= d (+ (to_real c) 0.5)))
>(assert (> a b))
>(check-sat)
>(get-model)
```

Getting values

- x + 2*y = 20 and x - y = 2 ?

```
> (set-option :produce-models true)
> (set-option :interactive-mode true)
> (set-logic QF_LIA)
> (declare-fun x () Int)
> (declare-fun y () Int)
> (assert (= (+ x (* 2 y)) 20))
> (assert (= (- x y) 2))
> (check-sat)
sat
> (get-value (x y))
((x 8)(y 6))
> (exit)
```

Getting Model

- x + 2*y = 20 and x - y = 2 ?

```
> (set-option :produce-models true)
> (set-option :interactive-mode true)
> (set-logic QF_LIA)
> (declare-fun x () Int)
> (declare-fun y () Int)
> (assert (= (+ x (* 2 y)) 20))
> (assert (= (- x y) 2))
> (check-sat)
sat
> (get-model)
(model
(define-fun x () Int 8)
(define-fun y () Int 6)
)
> (exit)
```

Assertion Levels

```
> (set-option :print-success false)
> (set-logic QF_LIA)
> (declare-fun x () Int)
> (declare-fun y () Int)
> (assert (= (+ x (* 2 y)) 20))
> (push 1)
> (assert (= (- x y) 2))
> (check-sat)
sat
> (pop 1)
> (push 1)
> (assert (= (- x y) 3))
> (check-sat)
unsat
> (pop 1)
> (exit)
```

Set Logic

```
> (set-logic AUFNIRA)
```

Can only appear one time in the context

- Boolean logics
  - QF_UF: Quantifier-Free, Uninterpreted Functions
- Logics with arithmetic
  - QF_LIA: QF_UF, Linear, Integer, Arithmetic
  - QF_NIA: QF_UF, Non-Linear, Integer, Arithmetic
  - QF_LRA: QF_UF
  - QF_AUFLIA: Array, QF_UF, Linear, Integer, Arithmetic,
  - AUFLIA : Allow Quantifier, Array, UF, Linear, Integer, Arithmetic
  - AUFLIRA: Allow Quantifier, Array, UF, Linear, Integer, Real, Arithmetic,
  - AUFNIRA: Allow Quantifier, Array, UF, Non-Linear, Integer, Real, Arithmetic
  - LRA: Allow Quantifier, Linear, Real, Arithmetic
- String Constraints
  - QF_S: Quantifier Free, String
  - S: String
- Bit-Vector
  - QF_BV

Set Logic

```
> (set-logic ALL_SUPPORTED)
```

Constants

```
>(declare-const b Int) ; syntax sugar for (declare-fun b () Int)
```

Uninterpreted Functions

```
>(declare-fun f (Int) Int)
>(declare-fun a () Int)
>(declare-const b Int)
>(assert (> a 20))
>(assert (> b a))
>(assert (= (f 10) 1))
>(check-sat)
>(get-model)
```

Propositional Logic

在這裡鍵入方程式。

```
(set-logic QF_UF)
(set-option :produce-models true)
(declare-const p Bool)
(declare-const q Bool)
(declare-const r Bool)
(define-fun c () Bool (=> (and (or p q) (=> q (not r))) (=> p r)))
(assert (not c))
(check-sat)
(get-model)
```

Array

```
(declare-const x Int)
(declare-const y Int)
(declare-const z Int)
(declare-const a1 (Array Int Int))
(declare-const a2 (Array Int Int))
(declare-const a3 (Array Int Int))
(assert (= (select a1 x) x))
(assert (= (store a1 x y) a1))
(check-sat)
(get-model)
```

Exercise

$$(x_2 \geq x_1) \wedge (x_1 - x_3 \geq x_2) \wedge (x_3 \geq 0) \wedge f(f(x_1) - f(x_2)) \neq f(x_3) \quad (10.1)$$

$$f(a[32], b[1]) = f(b[32], a[1]) \wedge a[32] = b[32] \quad (10.2)$$

Exercise

$$\begin{aligned}(f(x_1,0) \geq x_3) \wedge (f(x_2,0) \leq x_3) \wedge \\ (x_1 \geq x_2) \wedge (x_2 \geq x_1) \wedge \\ (x_3 - f(x_1,0) \geq 1)\,,\end{aligned} \tag{10.12}$$

Exercise

$$(x_2 \geq x_1) \wedge (x_1 - x_3 \geq x_2) \wedge (x_3 \geq 0) \wedge (f(f(x_1) - f(x_2)) \neq f(x_3)) \ . \quad (10.14)$$

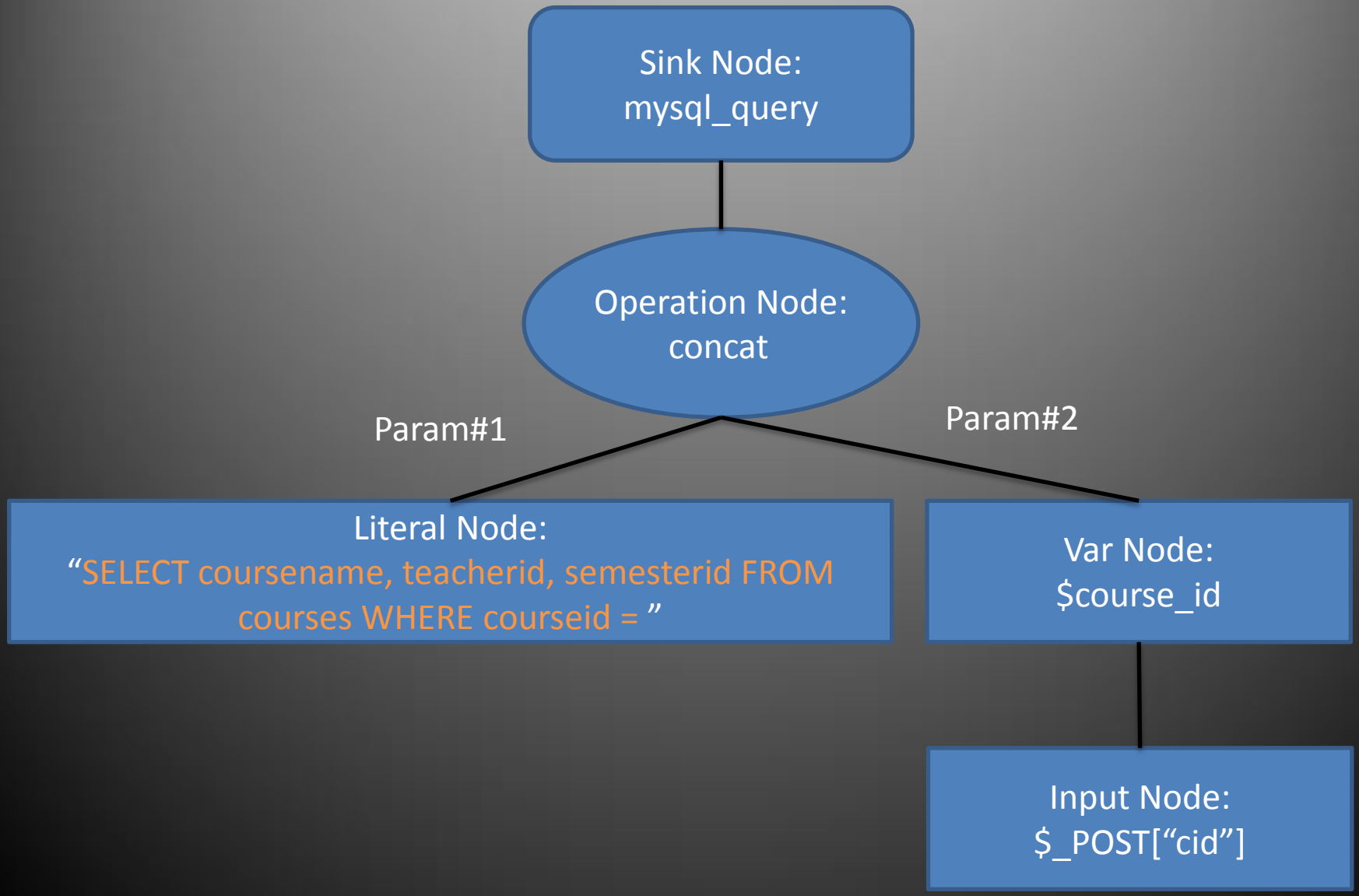
Exercise

$$(1 \leq x) \wedge (x \leq 2) \wedge p(x) \wedge \neg p(1) \wedge \neg p(2) \,, \qquad (10.16)$$

Case1: PHP Code

```
<?PHP
   $course_id = $_POST["cid"];
   $query = mysql_query("SELECT coursename, teacherid, semesterid FROM courses WHERE courseid = $course_id");
   while($class = mysql_fetch_row($query))
   {
        print("<h1>".$class[0]."</h1>");
   }
?>
```

Case1: Dependency Graph



String Constraint

- http://cvc4.cs.nyu.edu/wiki/Strings

Case2: PHP Code

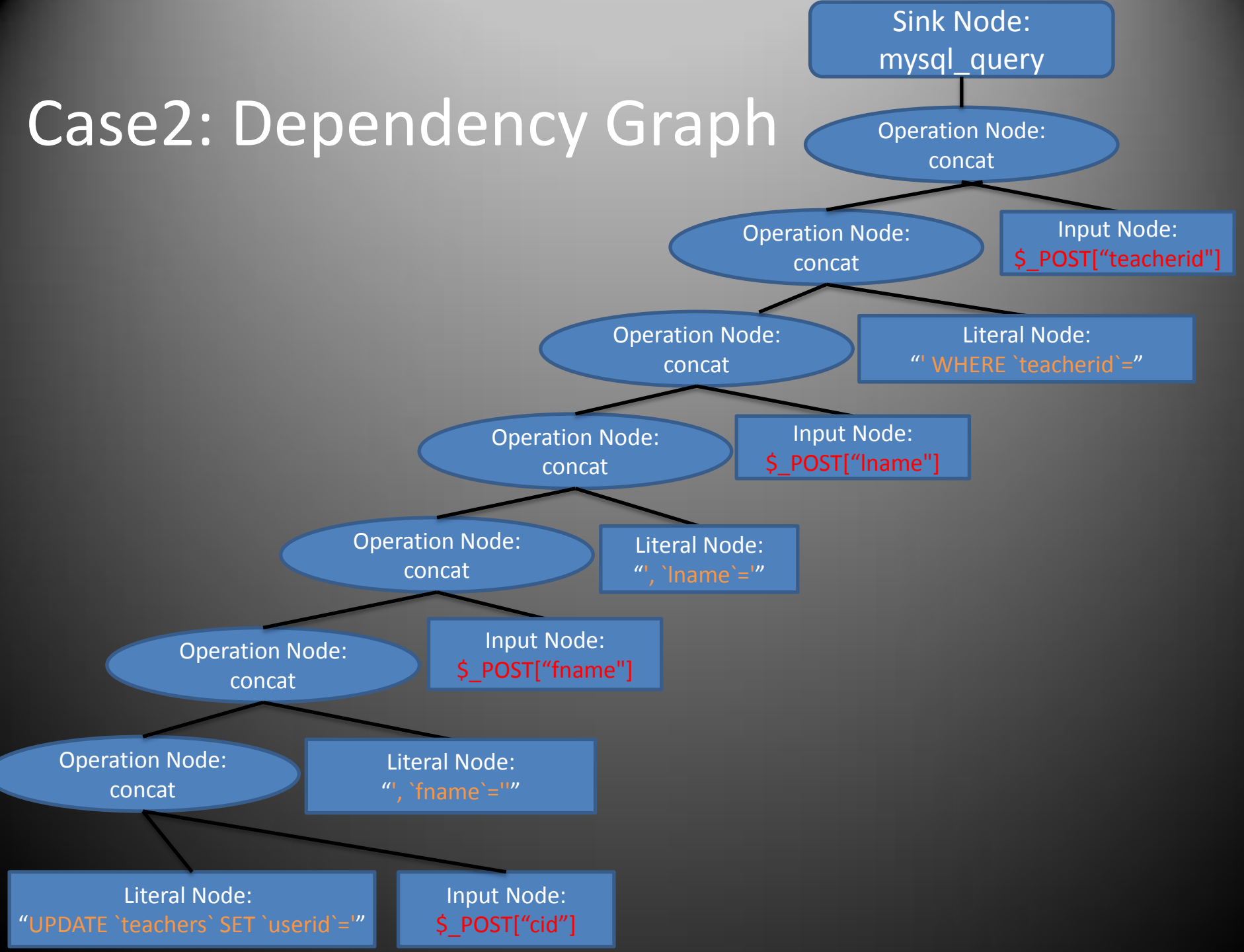
```
<?PHP

   $query = mysql_query("UPDATE `teachers` SET `userid`='".$_POST["username"]."',
`fname`='".$_POST["fname"]."', `lname`='".$_POST["lname"]."' WHERE
`teacherid`=".$_POST["teacherid"]);

   //…

?>
```

Case2: Dependency Graph



Case3: PHP

```php
<?PHP

 $a = $_POST["a"];
 $b = $_POST["b"];
 $statusCode = $_POST["statusCode"];
 $sql = mysql_query("SELECT * FROM tblCourse ");
 if($a+$b<3)
 {
    $sql = $sql." WHERE statusCode > ".$statusCode;
 }
 else
 {
    $sql = $sql." WHERE statusCode < ".$statusCode;
 }
 $query = mysql_query($sql);
 //....
?>
```

Case3: Dependency Graph