Formale Techniken der Software-Entwicklung

Übung 2

Christian Kroiß

25. April 2014

Transformation in CNF

```
function CNF ($\phi$) :
/* precondition: $\phi$ implication free and in NNF */
/* postcondition: CNF ($\phi$) computes an equivalent CNF for $\phi$ */
begin function
    case
        $\phi$ is a literal :  return $\phi$
        $\phi$ is $\phi_1 \wedge \phi_2$ :  return CNF ($\phi_1$) $\wedge$ CNF ($\phi_2$)
        $\phi$ is $\phi_1 \vee \phi_2$ :  return DISTR (CNF ($\phi_1$), CNF ($\phi_2$))
    end case
end function
```

Transformation in CNF (2)

```
function DISTR ($\eta_1, \eta_2$):
/* precondition: $\eta_1$ and $\eta_2$ are in CNF */
/* postcondition: DISTR ($\eta_1, \eta_2$) computes a CNF for $\eta_1 \vee \eta_2$ */
begin function
  case
    $\eta_1$ is $\eta_{11} \wedge \eta_{12}$:  return DISTR ($\eta_{11}, \eta_2$) $\wedge$ DISTR ($\eta_{12}, \eta_2$)
    $\eta_2$ is $\eta_{21} \wedge \eta_{22}$:  return DISTR ($\eta_1, \eta_{21}$) $\wedge$ DISTR ($\eta_1, \eta_{22}$)
    otherwise (= no conjunctions):  return $\eta_1 \vee \eta_2$
  end case
end function
```

Transformation in CNF (3)

```
function NNF ($\phi$):
/* precondition: $\phi$ is implication free */
/* postcondition: NNF ($\phi$) computes a NNF for $\phi$ */
begin function
  case
    $\phi$ is a literal: return $\phi$
    $\phi$ is $\neg\neg\phi_1$: return NNF ($\phi_1$)
    $\phi$ is $\phi_1 \wedge \phi_2$: return NNF ($\phi_1$) $\wedge$ NNF ($\phi_2$)
    $\phi$ is $\phi_1 \vee \phi_2$: return NNF ($\phi_1$) $\vee$ NNF ($\phi_2$)
    $\phi$ is $\neg(\phi_1 \wedge \phi_2)$: return NNF ($\neg\phi_1$) $\vee$ NNF ($\neg\phi_2$)
    $\phi$ is $\neg(\phi_1 \vee \phi_2)$: return NNF ($\neg\phi_1$) $\wedge$ NNF ($\neg\phi_2$)
  end case
end function
```

DPLL-Algorithmus (rekursive Version)

```
Function DPLL-recursive($F$, $\rho$)
Input: $F$: CNF Formula, $\rho$: initially empty partial assignment
Output: UNSAT, or an assignment satisfying $F$
begin
    $(F, \rho) \leftarrow$ UnitPropagate($F$, $\rho$)
    if $F$ contains the empty clause then return UNSAT
    if $F$ has no clauses left then
        Output $\rho$
        return SAT;
    $l \leftarrow$ a literal not assigned by $\rho$
    if DPLL-recursive($F|_l$, $\rho \cup \{l\}$) $=$ SAT then return SAT
    return DPLL-recursive($F|_{\neg l}$, $\rho \cup \{\neg l\}$)
end
```

DPLL-Algorithmus (rekursive Version)

```
Function UnitPropagate(F, $\rho$)
begin
    while $F$ contains no empty clause but has a unit clause $x$ do
        $F \leftarrow F|_x$
        $\rho \leftarrow \rho \cup \{x\}$
    end
    return $(F, \rho)$
end
```