$\{A\}\ p\ \{B\}$ The partial correctness assertion does not require p to terminate.

$[A]\ p\ [B]$ The total correctness assertion requires p to terminate.

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \qquad \sum_{k=1}^{n-1} k = \frac{n^2-n}{2}$$

$\vdash \{A\}\ p\ \{B\}$ ... triple $\{A\}\ p\ \{B\}$ is provable using Hoare rules

$\models \{A\}\ p\ \{B\}$ ... triple $\{A\}\ p\ \{B\}$ is valid

Hoare Logic is Sound: if $\{A\}\ p\ \{B\}$ is provable using Hoare rules, then $\{A\}\ p\ \{B\}$ is valid
(if $\vdash \{A\}\ p\ \{B\}$ then $\models \{A\}\ p\ \{B\}$)

3 / 54

- Soundness: if $\vdash \{A\}\ p\ \{B\}$ using Hoare rules then $\models \{A\}\ p\ \{B\}$

  "if a Hoare triple is proved to be valid using Hoare rules, then it is a valid Hoare triple."

- Completeness: if $\models \{A\}\ p\ \{B\}$ then $\vdash \{A\}\ p\ \{B\}$ using Hoare rules.

  "any valid Hoare triple can be proved to be valid using Hoare rules"

Hoare rule of consequence:

Strengthen precon Weaken postcon

$$\frac{A \Rightarrow A' \quad \{A'\}\ p\ \{B'\} \quad B' \Rightarrow B}{\{A\}\ p\ \{B\}}$$

Rule for Assignment

$$\overline{\{B[x/a]\}\ x := a\ \{B\}} \qquad \overline{\{A\}\ \textbf{skip}\ \{A\}} \qquad \overline{\{\text{true}\}\ \textbf{abort}\ \{B\}} \qquad [\text{false}]\ \textbf{abort}\ [B]$$

$$\frac{\{A\}\ p_1\ \{C\} \quad \{C\}\ p_2\ \{B\}}{\{A\}\ p_1; p_2\ \{B\}} \qquad \frac{\{A \wedge b\}\ p_1\ \{B\} \quad \{A \wedge \neg b\}\ p_2\ \{B\}}{\{A\}\ \textbf{if}\ b\ \textbf{then}\ p_1\ \textbf{else}\ p_2\ \{B\}}$$

Rule of sequence

$$\frac{\{I \wedge b\}\ p\ \{I\}}{\{I\}\ \textbf{while}\ b\ \textbf{do}\ p\ \textbf{od}\ \{I \wedge \neg b\}}$$

$$\frac{[A \wedge b \wedge t = t_0]\ p\ [A \wedge t < t_0] \quad A \wedge b \Rightarrow t \geq 0}{[A]\ \textbf{while}\ b\ \textbf{do}\ p\ \textbf{od}\ [A \wedge \neg b]}$$

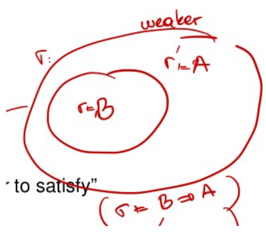
where A is an inductive loop invariant
$t \in$ AExp is a loop variant:
- t positive before each iteration
- t decreases strictly with each iteration
- t_0 is (fresh) auxiliary variable, storing value of t before iteration

- A is weaker than B iff $B \Rightarrow A$.

  "weaker" = "more states satisfying it" = "easier to satisfy"

- A is stronger than B iff $A \Rightarrow B$.

  "stronger" = "less states satisfying it" = "harder to satisfy"



$\text{wlp}(x := a, B) = B[x/a]$

$\text{wlp}(\textbf{skip}, B) = B$

$\text{wlp}(\textbf{abort}, B) = \text{true}$ $\quad \text{wp}(\textbf{abort}, B) = \text{false}$

$\text{wlp}(p_1; p_2, B) = \text{wlp}(p_1, \text{wlp}(p_2, B))$

$\text{wlp}(\textbf{if}\ b\ \textbf{then}\ p_1\ \textbf{else}\ p_2, B) = (b \Rightarrow \text{wlp}(p_1, B)\ \wedge\ \neg b \Rightarrow \text{wlp}(p_2, B))$

$\text{wlp}(\textbf{while}\ b\ \textbf{do}\ p\ \textbf{od}, B) = I$,

where I is an inductive loop invariant.

Theorem: $\{A\}\ p\ \{B\}$ is valid if $\text{VC}(p, B)\ \wedge\ (A \Rightarrow \text{wlp}(p, B))$

(1) Division

wlp

```
{x ≥ 0 ∧ y > 0}
quo := 0;  rem := x;
while y ≤ rem do
   rem := rem − y;  quo := quo + 1
od
{quo ∗ y + rem = x ∧ 0 ≤ rem < y}
```

Invariant:

$quo * y + rem = x\ \wedge$
$0 \leq rem \wedge 0 < y \wedge x \geq 0$

inv. $x = y(y-1)$

(2) Multiplication:

Ex!

```
{n > 0}
x := 0; y := 0;
while y < n do
   x := x + 2 ∗ y;  y := y + 1
od
{y = n}
```

$x[n+1] = x[n] + 2y[n]$

$y[n+1] = y[n] + 1$

Invariant? $\quad y[n] = n$

$\Updownarrow$

$x[n+1] = x[n] + 2 \cdot n$

$x[n] = x[n-1] + 2 \cdot (n-1)$

Try Dafny for proving the correctness assertions above!

$x[n] = n(n-1)$ $\quad x[n] = x[0] + 2 \cdot \frac{(n-1)n}{2}$ $\quad \Leftarrow \quad x[n] = x[0] + 2 \cdot \sum_{i=1}^{n-1} i$

57 / 67

$\text{VC}(x := a, B) = \text{true}$

$\text{VC}(\textbf{skip}, B) = \text{true}$

$\text{VC}(\textbf{abort}, B) = \text{true}$

VC of a sequence of assignments is "true"

$\text{VC}(p_1; p_2, B) = \text{VC}(p_2, B) \wedge \text{VC}(p_1, \text{wlp}(p_2, B))$

$\text{VC}(\textbf{if}\ b\ \textbf{then}\ p_1\ \textbf{else}\ p_2, B) = \text{VC}(p_1, B) \wedge \text{VC}(p_2, B)$

$\text{VC}(\textbf{while}\ b\ \textbf{do}\ q\ \textbf{od}, B) = (I \wedge \neg b) \Rightarrow B$
$\quad \wedge$
$\quad (I \wedge b) \Rightarrow \text{wlp}(q, I)$
$\quad \wedge$
$\quad \text{VC}(q, I)$.

$\text{VC}(\textbf{while}\ b\ \textbf{do}\ q\ \textbf{od}, B) = (I \wedge \neg b) \Rightarrow B$
$\quad \wedge$
$\quad (I \wedge b) \Rightarrow t \geq 0$
$\quad \wedge$
$\quad (I \wedge b \wedge t = t_0) \Rightarrow \text{wp}(q, I \wedge t < t_0)$
$\quad \wedge$
$\quad \text{VC}(q, I \wedge t < t_0)$

where I an inductive loop invariant, t is a loop variant.

Consider the loop **while** b **do** p **od**.

A loop invariant A:
- holds after each iteration of the loop.

An inductive loop invariant A:
- holds before and after each iteration of the loop. That is: $\{A \wedge b\}\ p\ \{A\}$.

Example: Consider the following IMP program:

```
x := 0; y := 0; n := 10;
while x < n do
      x := x + 1; y := y + x
od
```

$\{x+1 > 5\}\ \underline{x} := \underline{\underline{x+1}}\ \{\underline{\underline{x > 5}}\}$

$[x > 5][x \leftarrow x+1]$

$x \leq n$ is inductive invariant iff $\models \{x \leq n \wedge x < n\}\ x := x+1; y := y+x\ \{x \leq n\}$

$$\cfrac{x \leq n \wedge x < n \Rightarrow x+1 \leq n \qquad \cfrac{\overline{\{x+1 \leq n\}\ x := x+1\ \{x \leq n\}} \quad \overline{\{x \leq n\}\ y := y+x\ \{x \leq n\}}}{\{x+1 \leq n\}\ x := x+1; y := y+x\ \{x \leq n\}}\ \text{seq}}{\{x \leq n \wedge x < n\}\ x := x+1; y := y+x\ \{x \leq n\}}\ \textit{conseq}$$

So, $x \leq n$ is inductive invariant.

$t \to z < 10 \to t = 10 - z$
$t \to i \geq 0 \wedge i = i - 1 \to t = i$
$t \to x \neq 0 \wedge x = x + 1 \to t = x$
$t \to x \neq y \wedge y > 0 \wedge x = x + 1 \to y - x = t$
$t \to y < n \wedge n = 10 \wedge y = y + 1 \to t = n - y$
$x \neq 0\ \&\ x = x - 1$ than $x \geq 0$
$x \neq y\ \&\ x = x + 1$ than $y \geq x$
$z > 0\ \&\ z = z - 1$ than $z \geq 0$
wenn n vorkommt als konstante in [k+1]
dann ist es kn in [k]

The Hoare triple is not valid. For example, let σ be a state with $\sigma(i) = 0$ and $\sigma(n) = -1$. Then (1) $\sigma \models i = 0$, (2) $\langle\sigma, \textbf{while}\ i < n\ \textbf{do}\ i := i + 1\ \textbf{od}\rangle \to \sigma$ (because the loop is executed zero times) and (3) $\sigma \not\models i = n$.

CTL can be seen as a logic that is based on the compound temporal operators **AX**, **EX**, **AF**, **EF**, **AG**, **EG**, **AU**, and **EU**. In fact, each of the operators can be expressed in terms of the operators **EX**, **EG**, and **EU**:

$$\begin{aligned}
\mathbf{AX}\,\varphi &\equiv \neg\mathbf{EX}\,\neg\varphi\\
\mathbf{EF}\,\varphi &\equiv \mathbf{E}[\mathit{True}\,\mathbf{U}\,\varphi]\\
\mathbf{AG}\,\varphi &\equiv \neg\mathbf{EF}\,\neg\varphi\\
\mathbf{AF}\,\varphi &\equiv \neg\mathbf{EG}\,\neg\varphi\\
\mathbf{A}[\varphi\,\mathbf{U}\,\psi] &\equiv \neg\mathbf{E}[\neg\psi\,\mathbf{U}(\neg\varphi\wedge\neg\psi)]\wedge\neg\mathbf{EG}\,\neg\psi
\end{aligned}$$

The logic LTL only uses path formulas. This means that we restrict CTL* to disallow path quantification.

E1 If $p \in AP$, then p is an LTL formula.

E2 If φ is an LTL formula, then $\neg\varphi$, $\mathbf{X}\,\varphi$, $\mathbf{F}\,\varphi$, and $\mathbf{G}\,\varphi$ are LTL formulas.

E3 If φ and ψ are LTL formulas, then $\varphi\wedge\psi$, $\varphi\vee\psi$, and $\varphi\,\mathbf{U}\,\psi$ are LTL formulas.

AFAGp ≡ ¬EG¬AGp ≡ ¬EGEF¬p

Subformular: wenn neg dann auch das pos (sonst nicht)

$\Phi_0 = p$

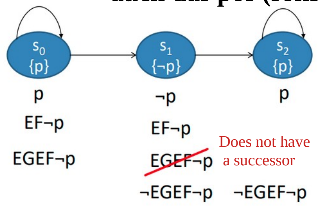
$\Psi_0 = \neg p$

$\Phi_1 = EF\neg p$

$\Psi_1 = EF\neg p$

$\Phi_2 = EGEF\neg p$

$\Psi_2 = \neg EGEF\neg p$



(1) $M,s \models p$ ⇔ $p \in L(s)$, for $p \in AP$

(2) $M,s \models \neg\varphi$ ⇔ $M,s \not\models \varphi$

(3) $M,s \models \varphi_1 \vee \varphi_2$ ⇔ $M,s \models \varphi_1$ or $M,s \models \varphi_2$

(4) $M,s \models \varphi_1 \wedge \varphi_2$ ⇔ $M,s \models \varphi_1$ and $M,s \models \varphi_2$

(5) $M,s \models \mathbf{E}\psi$ ⇔ there is a path π starting at state s such that $M,\pi \models \psi$

(6) $M,s \models \mathbf{A}\psi$ ⇔ for every path π starting at state s we have $M,\pi \models \psi$

(7) $M,\pi \models \varphi$ ⇔ s is the first state on path π and $M,s \models \varphi$

(8) $M,\pi \models \neg\psi$ ⇔ $M,\pi \not\models \psi$

(9) $M,\pi \models \psi_1 \vee \psi_2$ ⇔ $M,\pi \models \psi_1$ or $M,\pi \models \psi_2$

(10) $M,\pi \models \psi_1 \wedge \psi_2$ ⇔ $M,\pi \models \psi_1$ and $M,\pi \models \psi_2$

(11) $M,\pi \models \mathbf{X}\psi$ ⇔ $M,\pi^1 \models \psi$

(12) $M,\pi \models \mathbf{F}\psi$ ⇔ there exists a $k \geq 0$ such that $M,\pi^k \models \psi$

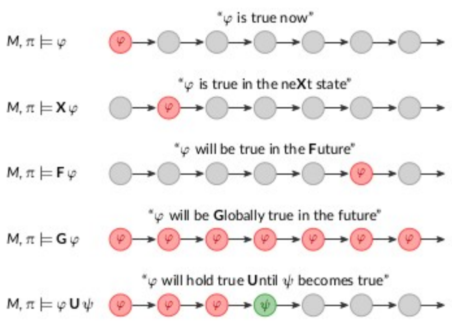
(13) $M,\pi \models \mathbf{G}\psi$ ⇔ for all $i \geq 0$, $M,\pi^i \models \psi$

(14) $M,\pi \models \psi_1\,\mathbf{U}\,\psi_2$ ⇔ there exists a $k \geq 0$ such that $M,\pi^k \models \psi_2$ and for all $0 \leq j < k$, $M,\pi^j \models \psi_1$

Finally, we define $\models$ over Kripke structure M as follows:

(15) $M \models \varphi$ ⇔ for all $s \in S_0$, $M,s \models \varphi$

(16) $M \models \psi$ ⇔ for all $s \in S_0$ and all paths π starting at s, $M,\pi \models \psi$


"φ is true now"
M, π ⊨ φ
"φ is true in the neXt state"
M, π ⊨ X φ
"φ will be true in the Future"
M, π ⊨ F φ
"φ will be Globally true in the future"
M, π ⊨ G φ
"φ will hold true Until ψ becomes true"
M, π ⊨ φ U ψ

- *If $M \equiv M'$ then for every CTL* formula φ,*

  $$M \models \varphi \Leftrightarrow M' \models \varphi$$

- *If $M' \succeq M$ then for every ACTL* formula φ,*

  $$M' \models \varphi \Rightarrow M \models \varphi$$

A formula to hold in a Kripke Structure it only needs to hold in the initial states.

Corollary

Let φ be an ACTL specification. If $M \preceq M'$ and $M' \models \varphi$ then $M \models \varphi$.

"M' simulates M" is denoted by $M' \succeq M$

Let $K_1 = (S_1, I_1, R_1, L_1)$ and $K_2 = (S_2, I_2, R_2, L_2)$.

$K_1 \geq K_2$: Assume $K_1 \geq K_2$ holds. Then, there is a simulation relation $H \subseteq S_2 \times S_1$. Now, consider state t_3 (note, t_3 is reachable from the initial state). In the simulation relation, the only possible corresponding state in K_1 would be s_4 (because of the labeling). Due to the transition (t_3, t_0), the pair (t_0, s_5) would have to be in the simulation relation as well. Then, (t_1, s_6) and (t_2, s_6) have to be in the simulation relation as well. But, there is no corresponding transition for (t_1, t_0) starting in s_6 and, therefore, (t_1, s_6) can't be in the simulation relation. Therefore, H can't exist and $K_1 \geq K_2$ does not hold.

GFGp ⇔ FGp

For the second direction, assume $K,\pi \models \mathbf{FG}p$. Consequently, $\exists x \geq 0 : K,\pi^x \models \mathbf{G}p$. In order to show the left hand side, we have to show that $\forall i \geq 0 : \exists j \geq i : K,\pi^j \models \mathbf{G}p$. Let $i \geq 0$.

- Case 1: $i \leq x$. In this case set $j := x$. By assumption, $K,\pi^x \models \mathbf{G}p$.
- Case 2: $i > x$. In this case set $j := i$. In order to show $K,\pi^i \models \mathbf{G}p$, let $k \geq i$. Given that $K,\pi^x \models \mathbf{G}p$ and $k \geq i > x$, it follows that $K,\pi^k \models p$.

Let K be a arb. Kripke structure and $\pi = s_0, s_1, ..$ an arb. path in K.
Assume $K,\pi \models (Ga \wedge Fb)$. Consequently, (1) $\forall i \geq 0 : K,\pi^i \models a$ and (2)$\exists x \geq 0 : K,\pi^x \models b$
By semantics of $\wedge$ and (1) and (2) we can conclude $K,\pi^x \models (b \wedge a)$ (*)
By semantics of $\vee$ and (1) we can conclude $\forall i \geq 0 : K,\pi^i \models (a \vee b)$ (**)
We need to show $K,\pi \models (a \vee b)U(a \wedge b)$:
- By (*) we have $\exists x \geq 0 : K,\pi^x \models (a \wedge b)$
- By (**) we have $\forall 0 \leq j \leq x : K,\pi^j \models (a \vee b)$
Combining the later two we have by semantics of U $K,\pi \models (a \vee b)U(a \wedge b)$

To show the validity of $\{1 \leq n\}\ p\ \{x = m * n\}$, we need to show that $VC(p, x = m * n) \wedge \big((1 \leq n) \Rightarrow wlp(p, x = m * n)\big)$.

We use the invariant $x = (y-1) * m \wedge y - 1 \leq n$, and compute the weakest liberal precondition $wlp(p, x = m * n)$ and the verification condition $VC(p, x = m * n)$.

By the definition of weakest preconditions we have

$$\begin{aligned}
wlp(p, x = m*n) &= wlp(x := 0, wlp(y := 1; \textbf{while } y \leq n \textbf{ do } q \textbf{ od}, x = m*n))\\
&= wlp(x := 0, wlp(y := 1, wlp(\textbf{while } y \leq n \textbf{ do } q \textbf{ od}, x = m*n)))\\
&= wlp(x := 0, wlp(y := 1, x = (y-1)*m \wedge y - 1 \leq n))\\
&= wlp(x := 0, x = (1-1)*m \wedge (1-1) \leq n)\\
&= 0 \leq n
\end{aligned}$$

As $1 \leq n$ implies $0 \leq n$, we have that the implication $(1 \leq n) \Rightarrow wlp(p, x = m*n)$ evaluates to true.

It remains to show that $VC(p, x = m*n)$ evaluates to true as well. We proceed by applying the definition of verification conditions and computing $VC(p, x = m*n)$.

$$\begin{aligned}
VC(p, x = m*n) &= VC(y := q; \textbf{while } y \leq n \textbf{ do } q \textbf{ od}, x = m*n)\\
&\quad \wedge VC(x := 0, wlp(y := 1; \textbf{while } y \leq n \textbf{ do } q \textbf{ od}, x = m*n))\\
&= VC(\textbf{while } y \leq n \textbf{ do } q \textbf{ od}, x = m*n) \wedge VC(y := 1, wlp(\textbf{while } y \leq n \textbf{ do } q \textbf{ od}, x = m*n)) \wedge \text{true}\\
&= VC(\textbf{while } y \leq n \textbf{ do } q \textbf{ od}, x = m*n) \wedge \text{true}
\end{aligned}$$

Thus, $VC(p, x = m*n) = VC(\textbf{while } y \leq n \textbf{ do } q \textbf{ od}, x = m*n)$. By the definition of verification condition for the while loop, we have

$x=1 \Rightarrow$ true   {true} abort {y>0}

{x=1} abort {y>0} ✓

$x=1 \Rightarrow 4=4$ {4=4} x:=4 {x=4}

? {x=1} x:=4 {x=4} ✓

VC(p, y>=n) = VC(x:=0;y:=1;while-loop, y>=n)

= VC(x:=0; y:=1, wp(while-loop, y>=n)) /\ VC(while-loop, y>=n)

= VC(x:=0; y:-1, loop-invariant) /\ VC(while-loop, y>=n)

= VC(while-loop, y>=n)

A Kripke structure is a five-tuple $M = (S, S_0, R, AP, L)$ where

- S is a (finite) set of states S
- $S_0 \subseteq S$ is the set of initial states
- $R \subseteq S \times S$ is a transition relation such that $\forall s \exists s' : (s, s') \in R$
- AP is some finite set of atomic propositions
- $L : S \to 2^{AP}$ is a is a function that labels each state with the set of those atomic propositions that are true in that state

We provide a polynomial time many-one reduction from 1-IN-3-SAT. Assume an arbitrary instance

$$\varphi = \bigwedge_{i=1}^{m} l_{i1} \vee l_{i2} \vee l_{i3}$$

of 1-IN-3-SAT over variables $V = \{v_1, \dots, v_n\}$. We construct an instance $\mathcal{C}$ of EXACT-HITTING-SET by setting

$$\mathcal{C} = \{\{v_j, \neg v_j\} \mid 1 \le j \le n\} \cup \ \{\{l_{i1}, l_{i2}, l_{i3}\} \mid 1 \le i \le n\}.$$

We show the correctness of the reduction by proving that

φ is a positive instance of 1-IN-3-SAT $\Longleftrightarrow \mathcal{C}$ is a positive instance of EHS

($\Longrightarrow$) Assume φ is a positive instance of 1-IN-3-SAT. Then there exists a truth assignment T over variables V that satisfies exactly one literal in each clause of φ. Let

$$S = \{v \mid T \text{ assigns true to } v\} \cup \{\neg v \mid T \text{ assigns false to } v\}.$$

We have to show $|S \cap C| = 1$, for each $C \in \mathcal{C}$. This holds for all sets $\{v_j, \neg v_j\}$ since T is a truth-assignment, i.e. assigns either true or false to v_j, and it holds for all sets $\{l_{i1}, l_{i2}, l_{i3}\}$ since T makes exactly one literal true in each clause $l_{i1} \vee l_{i2} \vee l_{i3}$.

($\Longleftarrow$) Assume $\mathcal{C}$ is a postive instance of EHS. Then there exists a set S such that $|S \cap C| = 1$, for each $C \in \mathcal{C}$. Define now an assignment T which assigns (for each j s.t. $1 \le j \le n$) true to variable v_j if $v_j \in S$ and false to v_j if $\neg v_j \in S$. In fact, T is a truth-assignment, because either v_j or $\neg v_j$ is in S by definition of EHS and sets $\{v_j, \neg v_j\}$ in $\mathcal{C}$. Moreover, T satisfies exactly one literal per clause in φ since S hits exactly one element in sets $\{l_{i1}, l_{i2}, l_{i3}\}$. Thus φ is a positive instance of 1-IN-3-SAT

Implication Law:	$(p \to q) \equiv (\neg p \vee q)$.
Distributive Laws:	$(p \vee (q \wedge r)) \equiv ((p \vee q) \wedge (p \vee r))$,
	$(p \wedge (q \vee r)) \equiv ((p \wedge q) \vee (p \wedge r))$.
de Morgan's Law:	$\neg(p \vee q) \equiv (\neg p \wedge \neg q)$
	$\neg(p \wedge q) \equiv (\neg p \vee \neg q)$.
Absorption Laws:	$(p \vee (p \wedge q)) \equiv p$,
	$(p \wedge (p \vee q)) \equiv p$.

We provide a polynomial time reduction from 3-COLORABILITY. Assume an arbitrary instance G of 3-COLORABILITY. We construct an instance (G, G') of 3COL-UNCOL by taking G' to be a fixed graph that is not 3-colorable, e.g. the complete graph K_4 (clique with 4 vertices). It remains to show the correctness of the reduction.

Assume G is a positive instance of 3-COLORABILITY. Then for the instance (G, G') the answer to the question in the definition of 3COL-UNCOL is "yes" since G' is not 3-colorable by definition. Hence, (G, G') is a positive instance of 3COL-UNCOL.

Assume (G, G') is a positive instance of 3COL-UNCOL. Then G is 3-colorable and G' is not 3-colorable, thus G is a positive instance of 3-COLORABILITY.

Membership: By providing a dual reduction from UNCOL to 3COL-UNCOL, we observe that 3COL-UNCOL is not only NP-hard but also coNP-hard. Thus unless NP=coNP the problem cannot be contained in NP.

NP-hardness: We provide a polynomial-time reduction from **3COL**. L
$G = (V, E)$ be an arbitrary instance of **3COL**. We construct an instance $G' = (V', E')$ of **3COL-NT** by setting $V' = \{v, v^1, v^2 \mid v \in V\}$ and $E' = E \cup \{[v, v^1], [v^1, v^2], [v, v^2]\}$. By definition, each such G' is a non-terminal graph thus the reduction yields the correct objects for the problem **3COL-NT**. We next show the correctness of the reduction.

Assume G is a positive instance of **3COL**. Then there exists a function μ from vertices in V to values in $\{0, 1, 2\}$ such that $\mu(v_1) \ne \mu(v_2)$ for any edge $[v_1, v_2] \in E$. Consider now $\mu^* : V' \to \{0, 1, 2\}$ defined as follows:

$$\text{for each } v \in V, \text{ let } \mu^*(v) = \mu(v) \text{ and } \mu^*(v^i) = (\mu(v) + i) \ mod\ 3.$$

We have to show that for any $[x, y] \in E'$, $\mu^*(x) \ne \mu^*(y)$. We have the following cases: (1) $x, y \in V$: then by construction and assumption that G is 3-colorable, $\mu^*(x) \ne \mu^*(y)$; otherwise $x, y \in \{v, v^1, v^2\}$ for some $v \in V$; by definition of μ^*, we then have $\mu^*(x) \ne \mu^*(y)$ as well. Hence, G' is 3-colorable and thus a positive instance of **3COL-NT**.

The following function f provides a polynomial-time many-one reduction from **IDS** to **SAT**: for a directed graph $G = (V, E)$, let

$$f(G) = \bigwedge_{(u,v)\in E} (\neg x_u \vee \neg x_v) \wedge \bigwedge_{v \in V} (x_v \vee \bigvee_{(u,v)\in E} x_u).$$

We show: G is a yes-instance of **IDS** $\Longleftrightarrow f(G)$ is a yes-instance of **SAT**.

($\Longrightarrow$): Suppose G is a yes-instance of **IDS**. I.e., there exists a set $S \subseteq V$ satisfying (1) and (2). We construct a truth assigment T as follows: $T(x_v) = true$ for $v \in S$; $T(x_v) = false$ for $v \in V \setminus S$.

We first show that $\varphi_1 = \bigwedge_{(u,v)\in E}(\neg x_u \vee \neg x_v)$ evaluates to *true* under T. Let $(u, v) \in E$. If $u \notin S$, $(\neg x_u \vee \neg x_v)$ evalutes to *true* under T since $\neg x_u$ is then true under T. If $u \in S$, by (1) $v \notin S$, and thus $(\neg x_u \vee \neg x_v)$ evalutes to *true* under T as well (since $\neg x_v$ is then true under T). Since this holds for all edges, φ_1 evalutes to *true* under T.

Now consider $\varphi_2 = \bigwedge_{v\in V}(x_v \vee \bigvee_{(u,v)\in E} x_u)$. Let $v \in V$. If $v \in S$ then $x_v \vee \bigvee_{(u,v)\in E} x_u$ evalutes to *true* under T thanks to the first disjunct x_v. If $v \notin S$, by (2) there exists $(u, v) \in E$ with $u \in S$. Then the corresponding disjunct in $\bigvee_{(u,v)\in E} x_u$ evalutes to *true* under T. Since this holds for all vertices, φ_2 evalutes to *true* under T.

($\Longleftarrow$): Suppose $f(G)$ is a yes-instance of **SAT**. Hence, there exists a truth-assignment T to the variables in $f(G)$ making this formula true. We construct a set $S = \{v \in V \mid T(x_v) = true\}$ and show that S satisfies (1) and (2).

(1): Towards a contradiction suppose (1) is violated by S, i.e. there exists $(u, v) \in E$ such that $u, v \in S$. By construction $T(x_u) = T(x_v) = true$. But then the conjunct $(\neg x_u \vee \neg x_v)$ in the first part of the formula $f(G)$ evalutes to false under T and thus T cannot make $f(G)$ true. Contradiction. Hence, S satisfies (1).

(2): Towards a contradiction suppose (2) is violated by S, i.e. there exists a vertex $v \in V$ such that $v \notin S$ and for all $(u, v) \in E$, $u \notin S$. By construction, we have $T(x_v) = false$ and, for all u with $(u, v) \in E$, $T(x_u) = false$. Since $f(G)$ has as a conjunct $x_v \vee \bigvee_{(u,v)\in E} x_u$, this subformula then evaluates to false under T and so does $f(G)$. Contradiction. Hence, S satisfies (2).

Our reduction shows NP-membership of **IDS**.

Provide a reduction from 2-**COLORABILITY** (2COL) to 3-**COLORABILITY** (3COL), and prove that your reduction is correct. Given this reduction and the fact that 3COL is NP-complete, what can be said about the complexity of **2COL**?

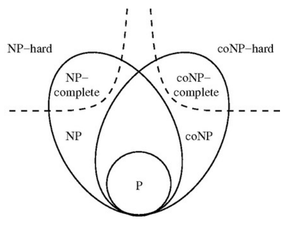
Assume an arbitrary instance $G = (V, E)$ of **2COL**. We create a new graph $G' = (V', E')$, where
- $V' = V \cup \{v'\}$ where v is a fresh vertex, and
- $E' = E \cup \{[v, v'] \mid v \in V\}$.

Intuitively, we add to G a new vertex v' and connect it to each original vertex of G.
We show that G is a positive instance of **2COL** iff G' is a positive instance of **3COL**.
- Suppose G can be properly colored with 2 colors (i.e., with 1 and 2). Then the existing coloring can be extended to G' by coloring v' with the additionally available color 3.
- Suppose G' can be properly colored with 3 colors. Since v' has an edge to every original node of G, all the original nodes must be properly colored with 2 colors only. It follows that G is 2-colorable.

ϕ^E Formeln:
dashed line = equality; solid line = inequality
1) umformen x5!=x6 in -e56 etc und aufzeichnen
2) remove all pure literals not part of
contradictional cycles (only 1 solid edge in cycle)
3) vereinfachte formel anschreiben + aufzeichnen
4) alle linien dick machen und linien einzeichnen
so dass es nur mehr 3ecke gibt
5) B_t = (e1 and e2 $\to$ e3) and (e2 and e3 $\to$ e1) and
(e3 and e1 $\to$ e2) and... (for each dreieck)
6) aufschreiben vereinfachteFormel and B_t (propositional logic)

complete graph =
alle knoten sind mit
allen verbunden

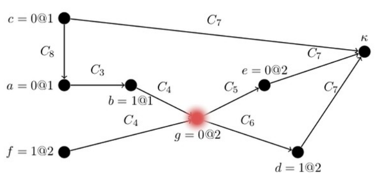
conected graph =
alle knoten sind
miteinander
verbunden



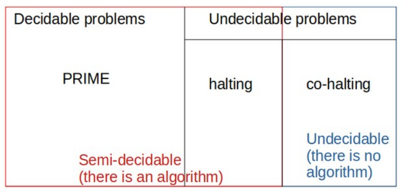
P	Q	$P \Rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

p	q	p↔q
T	T	T
T	F	F
F	T	F
F	F	T

- prove both directions; 4 different options:
  - pos. instance of $A \Rightarrow$ pos. instance of B;
    pos. instance of $B \Rightarrow$ pos. instance of A;
  - pos. instance of $A \Rightarrow$ pos. instance of B;
    neg. instance of $A \Rightarrow$ neg. instance of B;
  - neg. instance of $B \Rightarrow$ neg. instance of A;
    pos. instance of $B \Rightarrow$ pos. instance of A;
  - neg. instance of $B \Rightarrow$ neg. instance of A;
    neg. instance of $A \Rightarrow$ neg. instance of B;



$$res(C_7, C_5, e) = (c \vee g \vee \neg d)$$
$$res(R_1, C_6, d) = (c \vee g \vee g)$$
$$fac(R_2) = (c \vee g)$$



dimacs - sat solver
1) Tabelle umwandeln: -1 4 0 wird zu c1: (!x1 v x4)
2) wenn da assigned "true" stetht dann 1@1 starten
3) aufzeichnen und neuen state starten wenn man
einen neuen pfeil von außen her braucht
4) UIP: letzer knoten über den alle pfade zum conflict gehen

(1) We provide a polynomial time reduction from HALTING. Assume an arbitrary instance (Π, I) of HALTING. We construct an instance (Π', I_1, I_2) of DIFFERENT RUNTIME by setting

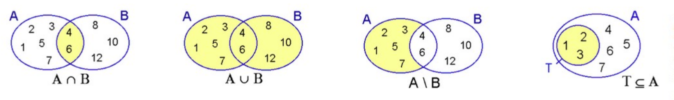
```
Boolean Π′ (String S)
    if $S = I_2$ then { while (true) do {} }
            else return $\Pi(S)$ // Π hardcoded
```

$I_1 = I$ and $I_2 \ne I$ an arbitrary string.
We show the correctness of the reduction, i.e. (Π, I) is a positive instance of HALTING iff (Π', I_1, I_2) is a positive instance of DIFFERENT RUNTIME.
Suppose (Π, I) is a positive instance of HALTING, then Π halts on I in a finite number of steps. By definition Π' also halts on $I_1 = I$ in a finite number of steps, but Π' does not terminate on I_2. Hence, (Π', I_1, I_2) is a positive instance of DIFFERENT RUNTIME.
Likewise, if (Π, I) is a negative instance of HALTING, then Π does not halt on I. By definition Π' then also does not halt on I_1 and thus terminates in the same number of steps than Π' does on I_2.

- $A = \{1, 4, 7\}$;
- $B = \{1, 4\}$;
- $C = \{4, 5, 7\}$;
- $D = \{3, 5, 6\}$;
- $E = \{2, 3, 6, 7\}$; and
- $F = \{2, 7\}$.

This is very similar to Exercise 6 of the Block 1 Exercises: Sample solutions (Page 21). Think of the definition of NP: a problem is in NP if there is a non-deterministic TM that (...). What happens when you apply this TM to instances of a special case of the problem?

Then $X^* = \{1, 2, 5\}$ is an exact hitting set, since each subset in S contains exactly one element in X^*,



$\to$ Proof by structural induction of the complexity of φ
Base case: Let $\varphi = p$ with $p \in AP'$
By semantics of ACTL we have $\forall s \in S_0 : M, s \models p$
As (by definition of M') $S' = S$, $S'_0 = S_0$, $R' = R$ and
$L'(s) = L(s) \wedge AP'$ we have $\forall s' \in S'_0 : M', s' \models p \Leftrightarrow M' \models p$
Same for $\varphi = \neg p$
Implication step:
Let $M \models \varphi \wedge \psi$ by semantics of $\wedge \to M \models \varphi$ and $M \models \psi$
By IH: $M' \models \varphi$ and $M' \models \psi$ by semantics of $\wedge \to M' \models \varphi \wedge \psi$

We provide a reduction from **co-HALTING**, which is known to be undecidable. Let (Π, I) be an arbitrary instance of **co-HALTING**. We build an instance Π' of **ALL-FALSE** by constructing Π' as follows:

```
String Π′ (Int n)
return Π_int(Π, I, n)   // Π and I are 'hard-coded' in Π′
```

To prove the correctness of the reduction we have to show:

(Π, I) is a positive instance of **co-HALTING** $\Leftrightarrow$ Π' is a positive instance of **ALL-FALSE**.

"$\Rightarrow$" Assume (Π, I) is a positive instance of **co-HALTING**, i.e. Π does not terminate on I. In particular, for any n, Π does not terminate on I within n steps. Hence, for any n, $\Pi_{int}(\Pi, I, n) = false$ by definition of Π_{int} and $\Pi'(n) = false$ by definition of Π'. That is, $\Pi'(n) = false$ for any natural number n. Thus Π' is a positive instance of **ALL-FALSE**.

"$\Leftarrow$" Assume Π' is a positive instance of **ALL-FALSE**, i.e. $\Pi'(n) = false$ for all natural numbers n. By definition of Π', $\Pi_{int}(\Pi, I, n) = false$ for all n. That is, there is no number n such that $\Pi_{int}(\Pi, I, n) = true$, i.e. such that Π terminates on I within n steps. Thus (Π, I) is a positive instance of **co-HALTING**.

(1) We provide a many-one reduction from HALTING. Assume an arbitrary instance (Π, I) of HALTING. We construct an instance (Π_1, Π_2, I') of ALOH by setting $\Pi_1 = \Pi$, Π_2 to a fixed program that runs into an infinite loop, and $I' = I$.
We show the correctness of the reduction, i.e. (Π, I) is a positive instance of HALTING iff (Π_1, Π_2, I') is a positive instance of ALOH.

($\Rightarrow$) Suppose (Π, I) is a positive instance of HALTING, i.e. $\Pi = \Pi_1$ halts on $I = I'$ in a finite number of steps. By definition, (Π_1, Π_2, I') is a positive instance of ALOH.

($\Leftarrow$) Likewise, if (Π, I) is a negative instance of HALTING, then $\Pi_1 = \Pi$ does not halt on $I = I'$. Since Π_2 does not halt on I' either by construction, (Π_1, Π_2, I') is thus a negative instance of ALOH.

Let $G = (V, E)$ be an arbitrary undirected graph, with $V = \{v_1, \dots, v_n\}$. Then the instance φ_G of **2-SAT** resulting from G is defined as follows: $\varphi_G = \bigwedge_{[v_i,v_j]\in E}(x_i \vee x_j) \wedge (\neg x_i \vee \neg x_j)$.
We show: G is a positive instance of **2COL** $\iff$ φ_G is a positive instance of **2-SAT**.

$\Longrightarrow$: Suppose G is a positive instance of **2COL**. Hence, there is a color assignment $f : V \to \{0, 1\}$ such that $f(v_i) \neq f(v_j)$ for all $[v_i, v_j] \in E$.
To show that φ_G is satisfiable, we define a truth assignment T as follows. For all $i \in \{1, \dots, n\}$,

$$T(x_i) = true \text{ if } f(v_i) = 1 \qquad T(x_i) = false \text{ if } f(v_i) = 0.$$

It remains to show that φ_G evaluates to *true* under T. Let $[v_i, v_j] \in E$. Since f is a proper 2-coloring of G, $T(x_i) \neq T(x_j)$.

- $T(x_i) = true$ and $T(x_j) = false$. Then trivially both clauses $(x_i \vee x_j)$ and $(\neg x_i \vee \neg x_j)$ evaluate to *true* under T.
- $T(x_i) = false$ and $T(x_j) = true$. Again, both clauses $(x_i \vee x_j)$ and $(\neg x_i \vee \neg x_j)$ evaluate to *true* under T.

$\Longleftarrow$: Suppose φ_G is positive instance of **2-SAT**. Then, there exists a truth assignment T such that $T(\varphi_G) = true$. We define a color assignment $f : V \to \{0, 1\}$ as follows (for $i \in \{1, \dots, n\}$):

$$f(v_i) = 1 \text{ if } T(x_i) = true \qquad f(v_i) = 0 \text{ if } T(x_i) = false.$$

It remains to show that f is a proper 2-coloring of G. Towards a contradiction, suppose this is not the case, i.e. there exists $[v_i, v_j] \in E$ with $f(v_i) = f(v_j) = \alpha$ ($\alpha \in \{0, 1\}$). We proceed with the argument for $\alpha = 1$ (the other case is analgous): by definition of f, we observe that $T(x_i) = T(x_j) = true$. But then, conjunct $(\neg x_i \vee \neg x_j)$ cannot be true under T. Consequently, also φ cannot be true under T. A contradiction to the assumption that $T(\varphi_G) = true$.

The reduction is defined as follows. Let (Π, I) be an arbitrary instance of **HALTING**. We build an instance (Π', n) of **REACHABLE-CODE** as follows. We let Π' be defined as

```
String Π′ (String S)
Π(I); // Π and I are hardcoded, S is ignored
return 0;
```

We let n be the line number of "**return** 0;" in Π'.

In other words, for an instance $x = (\Pi, I)$, the instance $R(x)$ resulting from the reduction is (Π', n). To prove the correctness of the reduction we have to show:

(Π, I) is a positive instance of **HALTING** $\Leftrightarrow$ (Π', n) is a positive instance of **REACHABLE-CODE**.

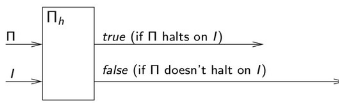
"$\Rightarrow$" Assume (Π, I) is a positive instance of **HALTING**, i.e. Π terminates on I. Then the call $\Pi(I)$ in program Π' terminates on any input S to Π'. Thus the statement "**return** 0;" is reached on any input to Π'. Hence, (Π', n) is a positive instance of **REACHABLE-CODE**.

"$\Leftarrow$" Assume (Π', n) is a positive instance of **REACHABLE-CODE**, i.e. Π' has an input S on which it reaches the line number n. Since the code of line n comes *after* the call $\Pi(I)$, it must be the case that Π terminates on I, i.e. (Π, I) is a positive instance of **HALTING**.

1-IN-3-SAT
INSTANCE: Propositional formula φ in 3-CNF
QUESTION: Does there exist a satisfying truth assignment T on φ, such that in each clause, exactly one literal is **true** in T?

SAME-OUTPUT
INSTANCE: A pair Π_1, Π_2 of programs that take a single string as input, an input string I.
QUESTION: Do Π_1 and Π_2 behave the same on input I? That is, Π_1 on I and Π_2 on I both return the same value or both do not terminate?

Assume there is a program Π_h such that:
- Π_h takes two strings as input:
  - Π (the source code of a program)
  - I (an input for the program Π)
- Π_h outputs:
  - *true* if Π terminates on I
  - *false* if Π does not terminate on I



$X \leq_P Y$ - Y mind so schwer wie X
$A \leq_P B \;\&\; B \in NP \Rightarrow A \in NP$
$A \leq_P B \;\&\; B \in P \Rightarrow A \in P$
$A \in NP \;\&\; B \in NP \;\&\; A \leq_P B \;\&\; A = NP-compl \Rightarrow B = NP-compl$
$A \leq_P B \;\&\; B \notin NP \;\&\; A = NP-comp \Rightarrow B = NP-hard$
NP-vollständig: NP-hard + $\in NP$
NP-hart: Jedes Problem in NP kann in poly Zeit auf dieses NP harte Problem reduziert werden.
NP-vollständige Probleme = schwerste Probleme in NP

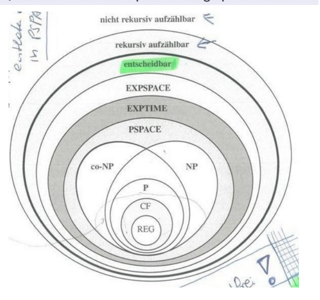
3-SAT
INSTANCE: Propositional formula φ in 3-CNF (i.e., CNF where each clause consists of exactly 3 literals).
QUESTION: Is φ satisfiable?

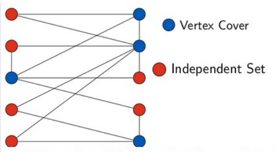
HAMILTON-CYCLE
INSTANCE: (directed or undirected) graph $G = (V, E)$
QUESTION: Does G have a *Hamilton cycle*?
i.e., a cycle visiting all vertices of G exactly once.

VALIDITY
INSTANCE: Propositional formula φ.
QUESTION: Is φ valid?

HAMILTON-PATH
INSTANCE: (directed or undirected) graph $G = (V, E)$
QUESTION: Does G have a *Hamilton path*?
i.e., a path visiting all vertices of G exactly once.

REACHABILITY
INSTANCE: A graph (V, E) and nodes $u, v \in V$.
QUESTION: Is there a path in the graph from u to v?





We know, that all threecolorable graphs stay threecolorable by removing vertices. By removing all nodes from $f(G) = (V', E')$ which were added during construction, we receive the previous graph which is then also threecolorable due to the given property. $G = (V, E)$ is therefor also a yes instance of **3-COL**

INDEPENDENT SET
INSTANCE: Undirected graph $G = (V, E)$ and integer K.
QUESTION: Does there exist an *independent set* I of size $\geq K$?
i.e., $I \subseteq V$, s.t. for all $i, j \in I$ the condition $[i, j] \notin E$ holds?

VERTEX COVER
INSTANCE: Undirected graph $G = (V, E)$ and integer K.
QUESTION: Does there exist a *vertex cover* N of size $\leq K$?
i.e., $N \subseteq V$, s.t. for all $[i, j] \in E$, either $i \in N$ or $j \in N$ or both.

CLIQUE
INSTANCE: Undirected graph $G = (V, E)$ and integer K.
QUESTION: Does there exist a *clique* C of size $\geq K$?
i.e., $C \subseteq V$, s.t. for all $i, j \in C$ with $i \neq j$, $[i, j] \in E$.

PROGRAM-EQUIVALENCE
INSTANCE: A pair Π_1, Π_2 of programs that take a single string as input.
QUESTION: Are Π_1 and Π_2 equivalent?
That is, is it true that for all inputs I, Π_1 on I and Π_2 on I both return the same value or both do not terminate?

CORRECTNESS
INSTANCE: Source code for a program Π that takes a string and outputs a string, and a pair of strings I_1, I_2.
QUESTION: Does Π return I_2 when run on input I_1?

Not-all-equal SAT (NAE at least one is true, and at least one is false
INSTANCE: Propositional formula φ in 3-CNF
QUESTION: Does there exist a satisfying truth assignment T on φ, such that the 3 literals in each clause do not have the same truth value?

EACHABLE-CODE
INSTANCE: Source code of a program Π, a number n of a line in Π.
QUESTION: Is there an input I for Π such that the run of Π on I will reach the code on line n?

(1) We can assume the availability of a decision procedure Π_{int} that does the following:

1. Π_{int} takes as input a program Π, a string I, and a natural number n.
2. Π_{int} emulates the first n steps of the run of Π on I. If Π terminates on I within n steps, then Π_{int} returns true. Otherwise, Π_{int} returns false.

We now provide a reduction from HALTING. Let (Π, I) be an arbitrary instance of HALTING. We construct an instance Π' of DIFFERENT OUTPUT as follows:

```
Boolean Π′ (Int n)
return Π_int(Π, I, n)   // Π and I are 'hard-coded' in Π′
```

If (Π, I) is positive instance of HALTING, then Π halts on I after n steps. Hence, $\Pi'(n) \neq \Pi'(n-1)$. It follows that Π' is a positive instance of DIFFERENT OUTPUT.
If (Π, I) is negative instance of HALTING, then $\Pi_{int}(\Pi, I, k)$ returns false for any k. Hence, $\Pi'(n_1) = \Pi'(n_2)$ for any pair of integers n_1, n_2. It follows that Π' is a negative instance of DIFFERENT OUTPUT.

SAT
INSTANCE: Propositional formula φ.
QUESTION: Is φ satisfiable?

The reduction is defined as follows. Let (Π, I_1, I_2) be an arbitrary instance of **CORRECTNESS**. We build an instance (Π', I') of **HALTING** by setting $I' = I_1$ and constructing Π' as follows:

```
String Π′ (String S)
OUT = Π(S); // Π is hardcoded in Π′
if OUT = I_2 then return 0
        else while True do {}
```

To prove the correctness of the reduction we have to show:

(Π, I_1, I_2) is a positive instance of **CORRECTNESS** $\Leftrightarrow$ (Π', I') is a positive instance of **HALTING**.

"$\Rightarrow$" Assume (Π, I_1, I_2) is a positive instance of **CORRECTNESS**, i.e. Π returns I_2 on input I_1. Then $OUT = I_2$ when I_1 is input to Π'. Then Π' terminates with output 0 on input I_1. Hence (Π', I') is a positive instance of **HALTING**.

"$\Leftarrow$" Assume (Π', I') is a positive instance of **HALTING**, i.e. Π' terminates on I'. Then the call $\Pi(S)$ in program Π' terminates on $S = I'$. This means that the "if" statement is reached by Π' on input I'. Since Π' terminates on I', it must be the case that $OUT = I_2$. Hence, we have the fact that Π returns I_2 on input I', where $I' = I_1$ by problem reduction, i.e. (Π', I_1, I_2) is a positive instance of **CORRECTNESS**.

We provide a polynomial-time reduction from **3SAT**.
Let $\psi = \bigwedge_{i=1}^{m}(l_{i1} \vee l_{i2} \vee l_{i3})$ be an arbitrary instance of **3SAT** We construct an instance φ of **RC3SAT** defined as

$$\varphi = \psi \wedge (x \vee x \vee x) \wedge (\neg x \vee \neg x \vee \neg x)$$

with x a fresh atom not occurring in ψ.
To prove the correctness of the reduction we have to show:

ψ is a positive instance of **3SAT** $\Leftrightarrow$ φ is a positive instance of **RC3SAT**.

"$\Rightarrow$" Let ψ be satisfiable and T be a satisfying truth-assignment for ψ. We extend T to T^* by additionally assinging x to true. We observe that φ without the last clause $(\neg x \vee \neg x \vee \neg x)$ evalutes to true under T^* (since T satisfies clauses $1..m$ of φ and setting x to true satisfies the remaining additional clause $x \vee x \vee x$). Hence φ is a positive instance of **RC3SAT**.

"$\Leftarrow$" Recall that φ consists of m clauses stemming from ψ and two additional clauses and assume φ is a positive instance of **RC3SAT**. Hence, there exists j $(1 \leq j \leq m+2)$ such that φ^{-j} is satisfiable. By definition of the additional clauses, j must be either $m+1$ or $m+2$ (otherwise φ^{-j} contains $(x \vee x \vee x) \wedge (\neg x \vee \neg x \vee \neg x)$ as subformula and would thus be unsatisfiable). It follows that φ^{-j} contains ψ as subformula. Hence, ψ is satisfiable and thus a positive instance of **3SAT**.

HALTING PROBLEM
INSTANCE: A (source code of) a SIMPLE program Π, an input string I.
QUESTION: Does the program Π terminate on input I?

A dominating set is a subset S of vertices such that every vertex not in S is adjacent to some vertex in S.

Clique: Subgraph wo alle knoten miteinander verbunden sind= vollständiger Teilgraph

3COL-UNCOL
INSTANCE: A pair (G_1, G_2) of undirected graphs.
QUESTION: Is it true that G_1 is 3-colorable and G_2 is not 3-colorable? I.e. is it the case that G_1 is a positive instance of **3-COLORABILITY** and G_2 is a negative instance of **3-COLORABILITY**?

EXACT HITTING SET (EHS)
INSTANCE: A collection $\mathcal{C}$ of sets of elements.
QUESTION: Does there exist a set S of elements, such that for each $C \in \mathcal{C}$, $|S \cap C| = 1$, i.e. each set in $\mathcal{C}$ contains exactly one element from S?

INDEPENDENT DOMINATING SET (IDS)
INSTANCE: A directed graph $G = (V, E)$.
QUESTION: Does there exists a set $S \subseteq V$ of vertices, such that
(1) for each $(u, v) \in E$, $\{u, v\} \not\subseteq S$;
(2) for each $v \in V$ either $v \in S$ or there exists an $(u, v) \in E$, such that $u \in S$.

ALL-HALTING
INSTANCE: A program Π that takes a single string as input.
QUESTION: Does Π halt on all input strings I?

k-COLORABILITY (for fixed value $k \geq 1$)
INSTANCE: Undirected graph $G = (V, E)$
QUESTION: Does G have a *k-coloring*? i.e., an assignment of one of k colors to each of the vertices in V such that any two vertices i, j connected by an edge $[i, j] \in E$ do not have the same color?