Definition: Semantikschema
Ein Semnatikschema ist ein Tupel $\langle R, I, [\![\cdot]\!] \rangle$ mit:

- R: Menge der Repräsentationen / Nachrichten
- I: Menge der Begriffe / Informationen
- $[\![\cdot]\!] \subseteq R \times I$: Semantik / Interpretation

Beispiel: Semantikschema für Unärdarstellung

- $R_u =_{\text{def}} \{|, ||, |||, \ldots\}$
- $I_u =_{\text{def}} \mathbb{N} = \{1, 2, 3, 4, \ldots\}$
- $[\![\cdot]\!]_u =_{\text{def}} \{\langle \underbrace{||\ldots|}_{k-\text{mal}}, k\rangle \mid k \in \mathbb{N}\}$

While Programme

Synatax: While Programme
Ausdrücke:
$a ::= n \mid x \mid (a + a) \mid (a * a)$
Bedingungen:
$b ::= true \mid false \mid (a == a) \mid (a < a) \mid (a <= a) \mid (!b) \mid (b \&\& b) \mid (b || b)$
Programme:
$S ::= x = a \mid skip \mid S; S \mid if(b)\{S\}else\{\} \mid while(b)\{S\}$

Wir legen für uns fest:

- Num für die Metavariable n,
- Var als die Variablen mit Metavariable x,
- $Aexp$ für die arithmetischen Ausdrücke mit Metavariable a,
- $Bexp$ für die Booleschen Ausdrücke mit Metavariable b,
- und Stm die die Anweisungen mit Metavariable S.
- $\Sigma =_{\text{def}} \{\sigma \mid \sigma : Var \to \mathbb{Z}\}$ als die (Speicher-)Zustände.

Semantik: Arithmetischer Ausdrücke
Die Semantik Funktion $[\![\cdot]\!]_A : Aexp \to (\Sigma \to \mathbb{Z})$ ist folgend definiert:

$$\begin{aligned}
[\![n]\!]_A(\sigma) &=_{\text{def}} [\![n]\!]_N \\
[\![x]\!]_A(\sigma) &=_{\text{def}} \sigma(x) \\
[\![(a_1 + a_2)]\!]_A(\sigma) &=_{\text{def}} \text{plus}([\![a_1]\!]_A(\sigma), [\![a_2]\!]_A(\sigma)) \\
[\![(a_1 * a_2)]\!]_A(\sigma) &=_{\text{def}} \text{mult}([\![a_1]\!]_A(\sigma), [\![a_2]\!]_A(\sigma))
\end{aligned}$$

Synatax: Erweiterte Boolescher Terme

$$\begin{aligned}
ABT &::= true \mid false \\
BT &::= ABT \mid (\neg BT) \mid (BT \wedge BT) \mid (BT \vee BT) \mid \\
&\quad (a < a) \mid (a = a) \mid (a \leq a) \mid (\forall y.BT) \mid (\exists y.BT)
\end{aligned}$$

Semantik: Boolescher Ausdrücke:
Die Funktion $[\![\cdot]\!]_B : Bexp \to (\Sigma \to \mathbb{B})$ ist folgend Definiert:

$$\begin{aligned}
[\![true]\!]_B(\sigma) &=_{\text{def}} \text{tt} \\
[\![false]\!]_B(\sigma) &=_{\text{def}} \text{ff} \\
[\![a_1 == a_2]\!]_B(\sigma) &=_{\text{def}} \text{tt}, \quad \text{falls eq}([\![a_1]\!]_A(\sigma), [\![a_2]\!]_A(\sigma)) \\
[\![a_1 == a_2]\!]_B(\sigma) &=_{\text{def}} \text{ff}, \quad \text{sonnst} \\
[\![\ !\text{b}\ ]\!]_B(\sigma) &=_{\text{def}} \text{neg}([\![b]\!]_B(\sigma)) \\
[\![\ \text{b}_1 \&\& \text{b}_2\ ]\!]_B(\sigma) &=_{\text{def}} \text{conj}([\![b_1]\!]_B(\sigma), [\![b_2]\!]_B(\sigma)) \\
[\![\ \text{b}_1 || \text{b}_2\ ]\!]_B(\sigma) &=_{\text{def}} \text{disj}([\![b_1]\!]_B(\sigma), [\![b_2]\!]_B(\sigma)) \\
[\![\forall y.b]\!]_B(\sigma) &=_{\text{def}} \text{tt} \Leftrightarrow \text{für alle } z \in \mathbb{Z} \text{ gilt } [\![b]\!]_B(\sigma\{z/y\}) = \text{tt} \\
[\![\exists y.b]\!]_B(\sigma) &=_{\text{def}} \text{tt} \Leftrightarrow \text{für min. ein } z \in \mathbb{Z} \text{ gilt } [\![b]\!]_B(\sigma\{z/y\}) = \text{tt}
\end{aligned}$$

... (andere Relatoren (z.B. $<, <=, \ldots$) analog)

Definition: Semantische Äquivalenz
Zwei Terme (Boolesch oder arithmetisch) t_1 und t_2 heißen **semantisch äquivalent**, geschrieben $t_1 \equiv t_2$, falls:

$$\forall \sigma \in \Sigma. [\![t_1]\!](\sigma) = [\![t_2]\!](\sigma)$$

Für $[\![a]\!]_A(\sigma) = [\![a']\!]_A(\sigma)$ oft auch kurz geschrieben: $[\![a]\!]_A = [\![a']\!]_A$.

Definition: Syntaktische Substitution
Die syntaktische Substitution für Terme aus T ist eine Abbildung:

$$\cdot[\cdot/\cdot] : T \times Aexp \times Var \to T$$

Für eine Konstate c, einen unären Operator u, einen binären Operator $\circ$, $a \in Aexp$, $t, t' \in T$ und Variablen $x, y \in Var$ induktiv definiert:

$$\begin{aligned} y[a/x] &=_{\text{def}} \begin{cases} a & \text{falls } y = x \\ y & \text{sonst} \end{cases} \\ c[a/x] &=_{\text{def}} c \\ (ut)[a/x] &=_{\text{def}} u(t[a/x]) \\ (t \circ t')[a/x] &=_{\text{def}} (t[a/x] \circ t'[a/x] \end{aligned}$$

Satz: Kompositionalität der Substitution
Set t ein Term und $a, a' \in Aexp$ mit $[\![a]\!]_A = [\![a']\!]_A$ und $x \in Var$ beliebig. Dann gilt:

$$[\![t[a/x]]\!] = [\![t[a'/x]]\!]$$

Lemma: Substitutionslemma
Sei t ein Term und $a \in Aexp$. Dann gilt für alle $\sigma \in \Sigma$:

$$[\![t[a/x]]\!](\sigma) = [\![t]\!](\sigma\{[\![a]\!]_A(\sigma)/x\})$$

Operationalle Semantik von While

Definition: Konfiguration
- Eine **nichtterminale Konfiguration** ist ein Tupel $\langle S, \sigma \rangle$, mit Programmfragment S und Zustand $\sigma \in \Sigma$.
- Eine **terminale Konfiguration** ist entweder ein Zustand σ oder $undef$.

Definition: Berechnungen
Wir legen folgende Begriffe fest:

- Ein **Berechnungsschritt** hat die Form: $\langle S, \sigma \rangle \Rightarrow \kappa$ mit $\kappa \in (Stm \times \Sigma) \cup \Sigma$
- Eine **Berechnungsfolge** zu einen Programm S zu einen Zustand $\sigma \in \Sigma$ ist
  - eine **endliche** Folge $\kappa_0, \ldots, \kappa_n$ von Konfigurationen mit $\kappa_0 = \langle S, \sigma \rangle$ und $k_i \Rightarrow k_{i+1}$ für alle $i \in \{0, \ldots, n-1\}$. Notation: $k_0 \Rightarrow^n k_n$
  - oder eine **unendliche** Folge von Konfigurationen mit $\kappa_0 = \langle S, \sigma \rangle$ und $k_i \Rightarrow k_{i+1}$ für alle $i \in \mathbb{N}$.
- Eine Berechnungsfolge heißt
  - **regulär terminierend**, wenn sie endlich ist und die letzte Konfiguration aus Σ ist.
  - **irregulär terminierend**, wenn sie endlich ist und die letzte Konfiguration $undef$ ist.
  - **divergierend**, falls sie unendlich ist.

Semantik: structural operational semantics (SOS)
Das Funktional $[\![\cdot]\!]_{SOS} : Stm \to (\Sigma - \to \Sigma)$ weißt jeder Anweisung eine *partielle* Funktion von den Zuständen in die Menge der Zustände zu. Allgemein definiert:

$$[\![S]\!]_{SOS}(\sigma) = \begin{cases} \sigma' & \text{falls } \langle S, \sigma \rangle \Rightarrow^* \sigma' \\ \text{undefiniert} & \text{sonst} \end{cases}$$

Semantik: Regeln für die SOS Semantik von While

$$[\mathbf{ass}_{\text{SOS}}] \quad \frac{}{\langle x = a, \sigma\rangle \Rightarrow \sigma\{[\![a]\!]_A(\sigma)/x\}}$$

$$[\mathbf{skip}_{\text{SOS}}] \quad \frac{}{\langle\ \text{skip}\ ,\ \sigma\ \rangle \Rightarrow \sigma}$$

$$[\mathbf{comp}^1_{\text{SOS}}] \quad \frac{\langle S_1, \sigma\rangle \Rightarrow \langle S_1', \sigma'\rangle}{\langle S_1; S_2, \sigma\rangle \Rightarrow \langle S_1'; S_2, \sigma'\rangle}$$

$$[\mathbf{comp}^2_{\text{SOS}}] \quad \frac{\langle S_1, \sigma\rangle \Rightarrow \sigma'}{\langle S_1; S_2, \sigma\rangle \Rightarrow \langle S_1', \sigma'\rangle}$$

$$[\mathbf{if}^{\text{tt}}_{\text{SOS}}] \quad \frac{}{\langle \text{if}(b)\{S_1\}\ \text{else}\ \{S_2\}, \sigma\rangle \Rightarrow \langle S_1, \sigma\rangle [\![b]\!](\sigma) = \text{tt}}$$

$$[\mathbf{while}_{\text{SOS}}] \quad \frac{}{\langle \text{while}(b)\{S\}, \sigma\rangle \Rightarrow \langle \text{if}(b)\{S; \text{while}(b)\{S\}\}\text{else}\{\text{skip}\}, \sigma\rangle [\![b]\!]}$$

Satz: Kompositionalität der strukturierten operationellen Semantik

Sei $[\![S_i]\!]_{SOS} = [\![S_i']\!]_{SOS}$ für bebliebige $S_i, S_i' \in Stm$ und $i \in \{1, 2\}$. Dann gilt:

1. $[\![S_i; S_2]\!]_{SOS} = [\![S_i'; S_2]\!]_{SOS}$
2. $[\![\text{if}(b)\{S_1\}\text{else}\{S_2\}]\!]_{SOS} = [\![\text{if}(b)\{S_1'\}\text{else}\{S_2\}]\!]_{SOS}$
3. $[\![\text{while}(b)\{S_1\}]\!]_{SOS} = [\![\text{while}(b)\{S_1'\}]\!]_{SOS}$

Flussgraphen

Definition: Flussgraph

Ein Flussgraph $G = (N, E, st, te)$, bestehend aus:

- Einer endlichen Menge von Knoten N
- einer endlichen Menge von gerichteter Kanten $E \subseteq N \times BA \times N$.
  Wobei $BA = \{\text{skip}\} \cup \{x = t \mid x \in \text{Var}, t \in \text{Aexp} \cup \{?\}\} \cup \text{Bexp}$ für die **Basisanweisung** steht, die jede Kante beschriften.
- Einem Startknoten $st \in N \wedge \nexists e \in E$ mit $e = (u, a, st), u \in N, a \in BA$
- Einem Terminierungsknoten $te \in N \wedge \nexists e \in E$ mit $e = (te, a, u), u \in N, a \in BA$

Definition: Pfade in Flussgraphen

- Endlicher Pfad π der Länge $k \leq 0$ in einem Flussgraphen G ist eine Folge von Kanten $(e_1, \ldots, e_k)$. Wobei $e_i = (u_i, b, v_i) \in E)$ für $i = 1, \ldots, k$ und $v_i = u_{i+1}$ für $i = 1, \ldots, k-1$
- Der Pfad der Länge $k = 0$ ist der leere Pfad und wird mit ϵ bezeichnet.
- Die Menge aller Pfade von u nach v wird mit $\text{Paths}[u, v]$ bezeichnet.

Semantik: Lokale Semantik von Flussgraphen

Die lokale Semantik von Flussgraphen wird durch die Funktion $[\![\cdot]\!]_{\text{FG}} : \text{BA} \to (\Sigma \to \mathcal{P}(\Sigma))$ festgelegt:

$$\begin{aligned} [\![\text{skip}]\!]_{\text{FG}}(\sigma) &= \{\sigma\} \\ [\![x = t]\!]_{\text{FG}}(\sigma) &= \{\sigma\{[\![t]\!]_A(\sigma)/x\}\} \\ [\![x = ?]\!]_{\text{FG}}(\sigma) &= \{\sigma\{z/x\} \mid z \in \mathbb{Z}\} \\ [\![b]\!]_{\text{FG}}(\sigma) &= \{\sigma \mid [\![b]\!]_A(\sigma) = \text{tt}\} \end{aligned}$$

Erweitert auf Kanten $e = (u, a, v) \in E$: $[\![e]\!]_{\text{FG}} =_{\text{def}} [\![a]\!]_{\text{FG}}$

Semantik: Pfadsemantik von Flussgraphen
Wir erweitern die Semantik induktiv auf Pfade:

$$\begin{aligned}[\![\epsilon]\!]_{\mathrm{FG}}(\Sigma') &= \Sigma' \\ [\![(e_1,\ldots,e_k)]\!]_{\mathrm{FG}}(\Sigma') &= [\![e_k]\!]_{\mathrm{FG}} \circ \ldots \circ [\![e_1]\!]_{\mathrm{FG}}(\Sigma')\end{aligned}$$

$$[\![e]\!]_{FG}(\Sigma') = \bigcup_{\sigma\in\Sigma'} [\![e]\!]_{FG}(\sigma)$$

Semantik: Flussgraphsemantik (Collecting-Semantik)
Die Semantik des Flussgraphen bezüglich einer Menge von Startzuständen wird durch die Collecting-Semantik ausgedrückt, die jedem Knoten die Menge der erreichbaren Zustände zuordnet:

$$[\![\cdot]\!]_{FG} : N \times \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$$
$$[\![u]\!]_{FG}(\Sigma') = \textstyle\bigcup_{\pi\in\mathrm{Paths}[st,u]} [\![\pi]\!]_{FG}(\Sigma')$$

Definition: Charakterisierung
Sei $P \in BT$. Dann ist die Charakteriserung von P definiert als:

$$Ch(P) =_{\mathrm{def}} \{\sigma \in \Sigma \mid [\![P]\!]_B(\sigma) = \mathrm{tt}\}$$

Lemma: Eigenschaften von $Ch(\cdot)$
Seien $P, Q \in BT$. Dann gilt:

1. $Ch(P \wedge Q) = Ch(P) \cap Ch(Q)$
2. $Ch(P \vee Q) = Ch(P) \cup Ch(Q)$
3. $Ch(\neg P) = \Sigma - Ch(P)$

Gültigkeit und partielle Korrektheit

Definition: Gültigkeit und partielle Korrektheit
Sei $G = (N, E, st, te)$ ein Flussgraph, $\Phi, pre, post \in BT$ und $u \in N$.

a) Φ heißt **gültig** an u unter Vorbedingung pre gdw.

$$[\![u]\!]_{FG}(Ch(pre)) \subseteq Ch(\Phi)$$

b) G heißt **partiell korrekt** bzgl. pre und $post$ gdw.
$post$ ist gültig unter Vorbedingung pre an te.

Definition: schwächste Vorbedingung
Sei $\Phi \in BT$. Wir definieren die schwächste Vorbedingung $wlp(a, \Phi) \in BT$ für eine Basisanweisung $a \in BA$ durch:

$$\begin{aligned}wlp(skip, \Phi) &= \Phi \\ wlp(x = t, \Phi) &= \Phi[t/x] \\ wlp(x =?, \Phi) &= (\forall z.\Phi[z/x]) \\ wlp(b, \Phi) &= (b \Rightarrow \Phi)\end{aligned}$$

Satz:
Sei $\Phi \in BT, a \in BA$. Dann gilt:

$$[\![a]\!]_{FG}(Ch(wlp(a, \Phi))) \subseteq Ch(\Phi)$$

Flyod

Fixpunktheorie

Definition: Partielle Ordnung
Eine Partielle Ordnung $(L, \sqsubseteq)$ liegt vor wenn die Menge L mit der binären Relation $\sqsubseteq \subseteq L \times L$ gilt:

- $\sqsubseteq$ ist reflexiv.
- $\sqsubseteq$ ist antisymmetrisch.
- $\sqsubseteq$ ist transitiv.

Für eine Teilmenge $X \subseteq L$ schreiben wir:

$\sqcup X$ für die kleinste obere Schranke (**Subpremum**), falls dieses existiert.

$\sqcap X$ für die größte obere Schranke (**Infimum**), falls dieses existiert.

Definition: Vollständiger Verband (complete lattice)
Eine partielle Ordnung $(L, \sqsubseteq)$, für die $\sqcup X$ für alle $x \subseteq L$ existiert.
Das eindeutig bestimmte kleine Element wird als $\bot = \sqcup\emptyset = \sqcap L$ festgelegt.
Das eindeutig bestimmte größte Element wird als $\top = \sqcup L = \sqcap\emptyset$ festgelegt.

Satz: Existenz von $\sqcap$ in einem Vollständigen Verband
Auch $\sqcap X$ existiert für alle $X \subseteq L : \sqcap X = \sqcup\{x \in L \mid x \sqsubseteq X\}$

Definition: Fixpunkt
Sei $(L, \sqsubseteq)$ eine partielle Ordnung und $f : L \to L$ **monoton** ($\forall x, y \in L : x \sqsubseteq y \Rightarrow f(x) \sqsubseteq f(y)$). Dann definieren wird für alle $x \in L$:

- x ist **Fixpunkt** von f gdw. $f(x) = x$.
- x ist **Prä-Fixpunkt** von f gdw. $f(x) \sqsubseteq x$
- x ist **Post-Fixpunkt** von f gdw. $f(x) \sqsupseteq x$.

Bemerkung:
$x \in L$ ist ein Fixpunkt von f gdw. x ist Prä- und Postfixpunkt von f.

Satz: Knaster-Tarski Fixpunkt Theorem
Jede monotone Funktion f auf einen vollständigen Verband L hat einen kleinen Fixpunkt $lfp(f)$ und einen größten Fixpunkt $gfp(f)$:

$$\begin{aligned} lfp(f) &= \sqcap\{x \in L \mid f(x) \sqsubseteq x\} \text{ (Infimum Prä-Fixpunkte)} \\ gfp(f) &= \sqcap\{x \in L \mid x \sqsubseteq f(x)\} \text{ (Supremum Post-Fixpunkte)} \end{aligned}$$

Definition: S
ei $(L, \sqsubseteq)$ eine partielle Ordnung.

- $K \subseteq L$ heißt **Kette** gwd. K ist total geordnet, d.h.:

$$\forall x, y \in K : x \sqsubseteq y \vee y \sqsubseteq x$$

- $f : L \to L$ ist **positiv Ketten-distributiv** (oder **postiv Ketten-stetig** gdw.

$$\forall \text{Kette } K : K \neq \emptyset \Rightarrow f(\sqcup K) = \sqcup f(K)$$

Satz: Knaster-Tarski-Kleene Fixpunkt Theorem
Sei $(L, \sqsubseteq)$ ein vollständiger Verband und $f : L \to L$ positiv Ketten-stetig. Dann ist

$$lfp(f) = \sqcup\{f^i(\bot) | i \in \mathbb{N}_0\}$$

wobei $f^0(\bot) = \bot, f^{i+1}(\bot) = f(f^i(\bot))$

Definition: Höhe einer Kette
Sei $(L, \sqsubseteq)$ ein partielle Ordnung und $k \in \mathbb{N}$.
$(L, \sqsubseteq)$ hat die **Höhe** k, falls k die maximale Länge einer Kette mit verschiedenen Elementen aus L ist.

Satz: Knaster-Tarski-Kleene Fixpunkt Theorem (endliche Version)
Sei $(L, \sqsubseteq)$ ein vollständiger Verband mit Höhe k und $f : L \to L$ monoton. Dann ist

$$lfp(f) = \sqcup\{f^i(\bot) | i \leq k\}$$

Satz: Knaster-Tarski-Kleene Fixpunkt Theorem (Erweiterte Fassung)
Für jede monotone Funktion f auf einem vollständigen Verband L ist die Menge der Fixpunkte $Fix(f)$ von f ein vollständiger Teilverband von L.
Bemerkung: I.a. ist für Fixpunkte d_1, d_2 das Supremum $d_1 \sqcup_L d_2$ kein Fixpunkt.

Definition: Monotones Datenflussanalyse(DFA) -Framkework
Ein Monotones DFA-Framework ist ein Tupel $((L, \sqsubseteq), F)$ mit:

- $(L, \sqsubseteq)$ ein vollständiger Verband. Die Elemente von L werden **data-flow facts** genannt.
- F ist Raum monotoner Transfer-Funktionen $f : L \to L$, derart dass
  - $Id \in F$
  - F ist abgeschlossen unter Komposition: $\forall f, g \in F : f \circ g \in F$

Definition: Instanz eines montonen DFA-Frameworks
Eine Instanz I eines montonen DFA-Frameworks $((L, \sqsubseteq), F)$ ist ein Tupel $I = (N, E, u_0, init)$. Mit folgenden Eigenschaften:

- (N, E) ist ein **(Kontrollfluss-) Graph** mit
  - endlicher Knoten N
  - jede Kante ist mit Transferfunktion $f_e \in F$ annotiert, d.h.:

$$E \subseteq N \times F \times N$$

- $u_0 \in N$ ist ein initialer Knoten, Jeder Knoten in N ist erreichbar von u_0.
- $init \in L$ ist eine **initiale Information**.

Beispiel: Framework und Instance für Analyse lebendiger Variablen

- Monotones Framework:
  - Vollständiger Verband:
$$(L, \sqsubseteq) = (\mathcal{P}(Var), \subseteq)$$
  - Raum von monotonen Transfer-Funktionen:
$$F = \{f : L \to L \mid \exists gen, kill \subseteq Var. \forall X \subseteq Var. f(X) = (X/kill) \cup gen\}$$
- Instanz $I = (N, E', te, Var)$ zu gebebenem Flussgraphen $G = (N, E, st, te)$:
  - Drehe die Kanten des Kontrollflussgraphen um (Rückwärtsanalyse).
  - Verändern der Annotation der Kanten:
$$f_e(X) = X/kill_e) \cup gen_e$$
    - $kill_e$ = Variable, der an e ein Wert zugewisen wird
    - gen_e = Variable, die (rechtsseitig) in e benutzt werden.
  - Initialer Knoten u_0 = Terminierungspunkt te
  - $init = Var$ (Menge der in G benutzen Variablen)
  - $E' = \{(v, f_e, u) \mid e = (u, a, v) \in E\}$

1 Modelchecking

Definition: Modelchecker
Ein Modelcheck ist ein Automatisches Verfahren das folgende Aussage entscheidet:

$$M \models \phi$$

Mit folgenden:

M	Modellstruktur
ϕ	(temporallogische) Formel
$\models$	der *modelliert*-Operator, oder simplel 'erfüllt'

1.1 Modellstruktur

Definition: Kripkestruktur

Definition: Transitionssystem

Definition: Kripke-Transitionssystem

Satz: Äquivalenz von Kripkestrukturen, Transitionssystemen und Kripke-T

1.2 Temporale Logiken

Synatax: Hennessy-Milner Logik (HML)
Wir definieren die Syntax in BNF:

$\phi := \text{true} \mid \neq \phi \mid \phi \vee \phi \mid <a> \phi$

$\phi := true \mid \neq \phi \mid$

Semantik: Hennessy-Milner Logik (HML)
Die Semantik von HML ist wird über einem Transitionssystem $T = (S, Act, \rightarrow)$ definiert. Wobei

S	Zustände
ϕ	Aktionen
$\rightarrow$	$\subseteq S \times Act \times S$

Dazu das Semantik-Funktional $[\![\cdot]\!]_T^{\text{HML}} : HTML \rightarrow \mathcal{P}(S)$ ist wie folgt definiert:

$$\begin{aligned} [\![\text{true}]\!]_T^{\text{HML}} &:= S \\ [\![\neq \phi]\!]_T^{\text{HML}} &:= S \backslash [\![\neq \phi]\!]_T^{\text{HML}} \\ [\![\phi_1 \vee \phi_2]\!]_T^{\text{HML}} &:= [\![\phi_1]\!]_T^{\text{HML}} \cup [\![\phi_2]\!]_T^{\text{HML}} \\ [\![< a > \phi]\!]_T^{\text{HML}} &:= \{P \mid \exists P'. P \rightarrow^a P' \wedge P' \in [\![\phi]\!]_T^{\text{HML}}\} \end{aligned}$$

Lemma: Duale HML Operatoren

$$\begin{aligned} \text{false} &:= \neg\text{true} \\ \phi_1 \wedge \phi_2 &:= \neg(\neg\phi_1 \vee \neg\phi_2) \\ [a]\phi &:= \neg < a > (\neg\phi) \\ \phi_1 \Rightarrow \phi_2 &:= \neg\phi_1 \vee \phi_2 \end{aligned}$$

Definition: $\models$ für Transitionsysteme und HMS
Wir definieren für ein Transitionssystem $T = (S, Act, \rightarrow, AP, I)$, einer HML Formel ϕ und einen Zustand $s \in S$:

$$s \models_{\text{HML}} \phi \Leftrightarrow s \in [\![\phi]\!]_T^{\text{HML}}$$

Synatax: Propositional Linear Time Logic (PLTL)
Wir definieren die PLTL Synatx in BNF:

$\phi ::= \text{p} \mid \neg \phi \mid \phi \vee \phi \mid \text{X } \phi \mid \phi \text{ U } \phi$

Semantik: Propositional Linear Time Logic
Die PLTL Semnatik wird auf dem Pfaden einer Krikpestruktur $K = (S, R, I)$ definiert.

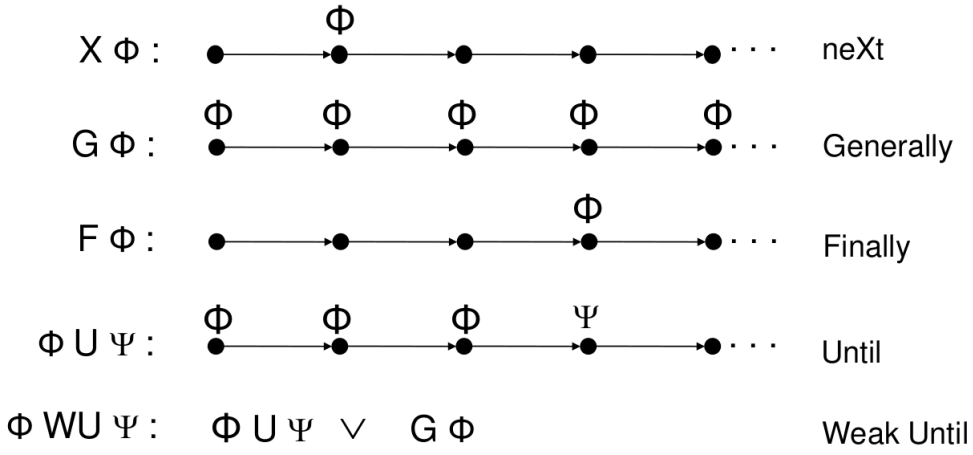
$\pi \models p$	falls $p \in I(\pi_0)$
$\pi \models \neg\phi$	falls $\pi \not\models \phi$
$\pi \models \phi_1 \vee \phi_2$	falls $\pi \models \phi_1$ oder $\pi \models \phi_2$
$\pi \models \text{X}\phi$	falls $\lvert\phi\rvert > 1$ und $\pi^1 \models \phi$
$\pi \models \phi\text{U}\psi$	falls es gibt $0 \leq k < \lvert\pi\rvert$. so dass $\pi^k \models \psi$ und für alle $0 \leq i < k$ gilt:$\pi^i \models \phi$

Definition: Abkürzungen für die PTLT
Wir führen die Synataktischen kurzschreibweisen ein:

$F(\phi)$	$\text{true}\text{U}\phi$
$G(\phi)$	$\neg F(\neg\phi)$
$\phi\text{WU}\psi$	$\phi\text{U}\psi \vee \text{G}\phi$

Dafür gibt es auch Semantik:

$\pi \models \text{F}\phi$	falls es gibt $0 \leq k < \lvert\pi$, so dass $\pi^k \models \phi$
$\pi \models \text{G}\phi$	falls für alle $0 \leq k < \lvert\pi$ gilt $\pi^k \models \phi$


X Φ :
neXt
G Φ :
Generally
F Φ :
Finally
Φ U Ψ :
Until
Φ WU Ψ :
Φ U Ψ ∨ G Φ
Weak Until