7.2 Syntax: Sprachmittel für Rechte

In diesem Kapitel führen wir Sprachmittel zur Spezifikation von Rechten ein.

Das sind zum einen Sprachmittel, um Rechte überschaubar(er) zu spezifizieren:

- implizite Rechte für Klassenmitglieder durch Klassennamen in Rechten sowie
- allgemeine Regeln mit -wenn nötig, mehrstufigen- Ausnahmen.

Außerdem beschreiben wir Sprachmittel, die Überlegungen des Entwurfsprozesses für spätere Änderungen bewahren:

- explizite Verbote und
- explizite Prioritäten.

7.2.1 Explizite Verbote

7.2.1.1 Unterscheidung zwischen Handlung und Recht

Oft gibt es keine syntaktische Unterscheidung zwischen einer Handlung und der Erlaubnis für diese Handlung.

In diesem Fall besteht ein Zugriffsrecht ebenfalls aus den drei Komponenten (Subjekt, Operation, Granul) und beschreibt, dass das Subjekt die Operation beim Granul ausführen darf, also die beschriebene Handlung erlaubt ist.

Damit einher geht, dass entweder nur Erlaubnisse oder nur Verbote explizit gemacht werden können.

Meistens werden nur die Erlaubnisse explizit ausgedrückt, die Verbote jedoch nur implizit (mit Hilfe der Meta-Regel "alles, was nicht ausdrücklich erlaubt ist, ist verboten").

Dann lässt sich jedoch syntaktisch auch kein Unterschied zwischen

"Handlung a ist verboten"

und

"Es ist irrelevant, ob Handlung a erlaubt oder verboten ist"

ausdrücken; in beiden Fällen würde für diese Handlung kein Recht spezifiziert.

In diesem Fall löscht jede später eingefügte Erlaubnis implizit ein Verbot, ohne dass klar ist, ob diese Änderung nur eine fehlende Spezifikation hinzufügt oder ein bewusst vorgesehenes Verbot löscht!

Wir möchten einerseits syntaktisch zwischen einer Handlung und einem Recht unterscheiden können;

andererseits können wir diese zusätzliche Unterscheidungskomponente gleichzeitig für eine Unterscheidung zwischen Erlaubnissen und Verboten benutzen.

7.2.1.2 Rechtekennung für Erlaubnisse und Verbote

Wir wollen daher nicht nur Erlaubnisse, sondern auch Verbote explizit modellieren.

Dadurch können wir die Verbote, die während der Entwurfsphase für nötig gehalten werden, dauerhaft machen:

Explizite Verbote können anschließend nicht einfach –mit oder ohne Absicht– ignoriert werden, indem einfach entsprechende Erlaubnisse eingefügt werden, sondern solche Rechteänderungen führen nun zu einem Rechtekonflikt (und einer entsprechenden Fehlermeldung).

Darüberhinaus werden wir explizite Verbote benötigen, um Ausnahmen beschreiben zu können.

Nicht zuletzt führen wir durch explizite Verbote Redundanz ein und erhalten damit die Möglichkeit, Entwurfsfehler zu entdecken.

Diese Methode ist vom Programmiersprachenentwurf (z.B. die Forderung, Namen zu deklarieren) gut bekannt.

Wir können dann nämlich insbesondere feststellen, ob für eine Handlung keine Rechtespezifikation oder sogar mehrere widersprüchliche Rechtespezifikationen vorliegen.

Wir ergänzen daher eine Handlung um eine weitere Komponente, eine **Rechtekennung**.

Z.B. erlaubt (Erlaubnis, a) die Handlung a, während (Verbot, a) die Handlung a verbietet.

Der Deutlichkeit halber verwenden wir für Rechtekennungen die Menge

TAG={Verbot, Erlaubnis}

statt

BOOLEAN={false, true},

benutzen aber auch für diese Menge die booleschen Operationen.

7.2.1.3 Explizite Rechte

Definition: Die Menge aller expliziten Rechte ist XR:=TAG×EA.
Ein **explizites Recht** ist also ein Tupel xr=(t, ea)=(t, (es,eo,eg)) TAG×EA.
Wir schreiben im Folgenden einfach (t,s,o,g) anstelle von (t, (s,o,g)).

In der (dreidimensionalen) Rechtematrix entspricht diese Verallgemeinerung folgender Vergrößerung des Wertebereichs: M: S×O×G → ({erlaubt, verboten})

Es können daher neben den Matrixwerten 'erlaubt' und 'verboten' auch alle anderen Werte der Potenzmenge dieser Grundmenge vorkommen, nämlich Ø (im Sinne von unvollständiger Spezifikation) und {erlaubt, verboten} (im Sinne von widersprüchlicher Spezifikation).

Bei der zweidimensionalen Matrix müssten wir jetzt z.B. neben einer **Erlaubnismatrix** zusätzlich eine **Verbotsmatrix** einführen und zur Erkennung unvollständiger oder widersprüchlicher Spezifikationen beide Matrizen miteinander vergleichen.

Die Menge aller expliziten Rechte soll unmittelbar die Frage beantworten:
"Wer darf was bei wem (nicht) tun?".

Explizite Rechte sind einfach zu verstehen, aber umständlich zu spezifizieren.
Daher führen wir mehrere Abkürzungsmöglichkeiten ein.

7.2.2 Mengen und Klassennamen in Rechten

Eine Möglichkeit zur Abkürzung erhalten wir durch die Nutzung der Objekt-Klassen- und Klassen-Oberklassen-Beziehungen.

Dazu erlauben wir syntaktisch Handlungen (und damit Klassennamen) statt elementarer Handlungen in der letzten Komponente eines Rechts.

Das Vorkommen eines **Klassennamens** in einem Recht bedeutet, dass das Recht implizit gültig ist für alle **Klassenmitglieder** dieser Klasse.

Außerdem berücksichtigen wir beim Vorkommen eines Klassennamens die **Klassenhierarchie**, und können so auch eine Vererbung des Rechts entlang der Klassenhierarchie ausdrücken.

Wir legen fest, dass das Vorkommen eines Klassennamens in einem Recht bedeutet, dass das Recht implizit gültig für alle Unterklassen ist.

Allerdings werden wir abhängig von der Rechtekennung z.T. unterschiedliche Klassenhierarchien verwenden.

Eine **Menge** von Objekten wird elementweise ausgewertet, z.B.
(Hautarzt, Diagnose, Haut) steht für
({catherine}, {untersuchen, röntgen}, {haut}) und schließlich für
{(catherine, untersuchen, haut), (catherine, röntgen, haut)}.

7.2.3 Mehrstufige Ausnahmen

7.2.3.1 Regeln und Ausnahmen

Eine andere Abkürzungsmöglichkeit besteht in der Spezifikation von **Ausnahmen**:

Viele Anwendungen können klarer (und kürzer) beschrieben werden mit Hilfe einer allgemeinen Regel und wenigen Ausnahmen zu dieser Regel.

Um allgemeine Regeln zu beschreiben, können wir vorteilhaft die Objekt-Klassen- und Klassen-Oberklassen-Beziehungen verwenden.

Wenn wir Ausnahmen als Spezifikationsmittel zulassen wollen, benötigen wir ein Sprachmittel, das kennzeichnet, was eine allgemeine Regel ist und was eine Ausnahme (zu dieser Regel) darstellt: **Prioritäten**.

Wann immer wir Ausnahmen benutzen wollen, sind wir gezwungen, ein Prioritätssystem einzuführen.

Für implizite Prioritäten sind verschiedene Varianten naheliegend:

1. Verbote haben stets Vorrang vor Erlaubnissen.

   Leider kann man dann keine Ausnahmen modellieren, wo der allgemeine Fall ein Verbot und die Ausnahme eine Erlaubnis ist!

2. Das speziellere Recht hat Vorrang vor dem allgemeineren Recht:

   Diese Variante modelliert den Ausnahmemechanismus unmittelbar, aber implizit.

   Die Wirkung von Rechteänderungen wird dabei aber kompliziert und wenig übersichtlich.

   Außerdem gibt es mehrere Möglichkeiten festzulegen, was "spezieller" sein soll:
   "spezielleres Recht" könnte z.B. sein das Recht mit dem spezielleren Subjekt, nur bei gleichem Subjekt: das mit der spezielleren Operation, nur bei gleicher Operation, das mit dem spezielleren Granul.

   Dieser Ansatz hat daher Schwierigkeiten mit Rechtemengen, wenn sich bzgl. verschiedener Komponenten unterschiedliche Ordnungsrelationen ergeben, z.B. in einem Recht das speziellere Granul, aber die allgemeinere Operation und in einem anderen Recht die speziellere Operation, aber das allgemeinere Granul vorkommt; z.B. ist bei der Rechtemenge

   {(Krankenschwester, Therapie, Gliedmaßen, Verbot),
   (Krankenschwester, injizieren, Körper, Erlaubnis)}

   nicht klar, ob die Handlung (Krankenschwester, injizieren, Gliedmaßen) erlaubt oder verboten ist.

3. Ein neueres Recht hat Vorrang vor älteren Rechten:

   Dies erfordert in der Implementierung eine zusätzliche Zeitkomponente; außerdem hätte dann das Löschen eines Rechtes und anschließendes Wiedereinfügen desselben Rechtes (z.B. als Versuch einer Wiederherstellung nach versehentlichem Löschen) u.U. gravierende semantische Auswirkungen.

7.2.3.2 Explizite Prioritäten

Nicht nur Verbote, sondern auch das Prioritätssystem sollte explizit sein.

Das zwingt den Rechteverwalter, über die Priorität nachzudenken und das Ergebnis explizit nachvollziehbar zu dokumentieren.
Damit gibt es die Möglichkeit, auch später noch Inkonsistenzen zu erkennen, die von der Entwurfsphase stammen (z.B. falsche Prioritätszuordnung).
Benutzt man dagegen implizite Prioritäten zwischen Erlaubnissen und Verboten, so gibt es kaum Möglichkeiten, derartige Inkonsistenzen zu erkennen.

Der hier vorgestellte Ansatz lässt eine **mehrstufige Ausnahmebehandlung** zu, da schon in Alltagssituationen eine mehrstufige Ausnahmebehandlung üblich ist.

So sind z.B. bei den Verkehrsregeln die gängigen Vorfahrtsregelungen (rechts-vor-links, Vorfahrtsschilder, Ampeln, Einsatzfahrzeuge mit Blaulicht und Sirene, Verkehrsregelung durch Polizisten) sowie die Überholregeln fünfstufig.

7.2.3.3 Spezifizierte Rechte

Formal führen wir daher eine weitere Komponente eines Rechts ein, eine **Priorität**.

Die Idee dahinter ist, dass sich unter allen Rechten mit der gleichen Handlung a dasjenige Recht mit der maximalen Priorität unabhängig von der Rechtekennung durchsetzt,
d.h. das Recht (Erlaubnis, 7, a) unterdrückt (Verbot, 5, a).

Definition: Sei PRIO eine Menge von Prioritäten.
Die Menge aller spezifizierten Rechte ist SR:=TAG×PRIO×A.
Ein **spezifiziertes Recht** ist also ein Tupel sr=(t,p,sa) TAG×PRIO×A.
Seine Komponenten sind eine Handlung, eine Rechtekennung und eine Priorität.
Die Menge PRIO sei (zumindest) partiell geordnet.

Zunächst verwenden wir linear geordnete Prioritäten, in den Beispielen natürliche Zahlen.
Später untersuchen wir dann partiell geordnete Prioritäten.

Bei der (dreidimensionalen) Rechtematrix entspricht diese Verallgemeinerung wieder einer Vergrößerung des Wertebereichs: M: S×O×G → ({erlaubt, verboten}×PRIO).

Nun können wir Ausnahmen modellieren:

Wir beschreiben eine allgemeine Regel, indem wir eine niedrige Priorität benutzen und recht allgemeine Klassen, um eine große Objektmenge zu bezeichnen.

Eine Ausnahme zu dieser Regel können wir dann spezifizieren, indem wir eine größere Priorität benutzen, die invertierte Rechtekennung und eine einschränkendere Bezeichnung für die Menge der Objekte.

Die Regeln

{(Verbot, 20, Krankenschwester, Therapie, Körper),
(Erlaubnis, 30, Krankenschwester, injizieren, Gliedmaßen)}

stellen fest, dass eine Krankenschwester im allgemeinen nicht therapieren darf, jedoch ausnahmsweise in die Gliedmaßen injizieren darf.

7.2.4 Beispiel

Beispiel: In unserem Medizinbeispiel seien folgende Rechte spezifiziert:

SR_1 = {	(Erlaubnis,	50,	Chirurg,	Med. Operation,	Innere Organe),
	(Verbot,	60,	hendrik,	Med. Operation,	herz),
	(Verbot,	20,	Arzt,	transplantieren,	Körper),
	(Erlaubnis,	10,	Arzt,	Therapie,	Körper),
	(Verbot,	20,	Zahnarzt,	Therapie,	Gliedmaßen),
	(Verbot,	20,	Zahnarzt,	Therapie,	Rumpf),
	(Erlaubnis,	10,	Krankenschwester,	Pflege,	Körper),
	(Verbot,	20,	Krankenschwester,	Therapie,	Körper),
	(Erlaubnis,	30,	Krankenschwester,	injizieren,	Gliedmaßen) }

7.3 Semantik von Rechtespezifikationen: Übersetzung in explizite Rechte

In diesem Kapitel wollen wir präzise die Semantik einer Menge von spezifizierten Rechten beschreiben, die dadurch unübersichtlich werden kann, dass die betroffenen Objektmengen sich überlappen können wie z.B. bei

{(Verbot, 20, Arzt, transplantieren, Körper),
(Erlaubnis, 50, Chirurg, Med. Operation, Innere Organe)}.

Die Bedeutung einer Menge von spezifizierten Rechten wird definiert durch Angabe eines anschaulichen Erzeugungs- oder Übersetzungsverfahrens in explizite Rechte.

Dieser Erzeugungsprozess besteht im wesentlichen aus drei Schritten:

- Auswertung der Klassen-Oberklassen-Beziehung,
- Auswertung der Objekt-Klassen-Beziehung,
- Auswertung der Prioritäten.

Außerdem können bei der Auswertung evtl. vorhandene Inkonsistenzen entdeckt werden.

Die ersten beiden Schritte berechnen die Menge der elementaren Handlungen, die von einer spezifizierten Handlung beschrieben werden.

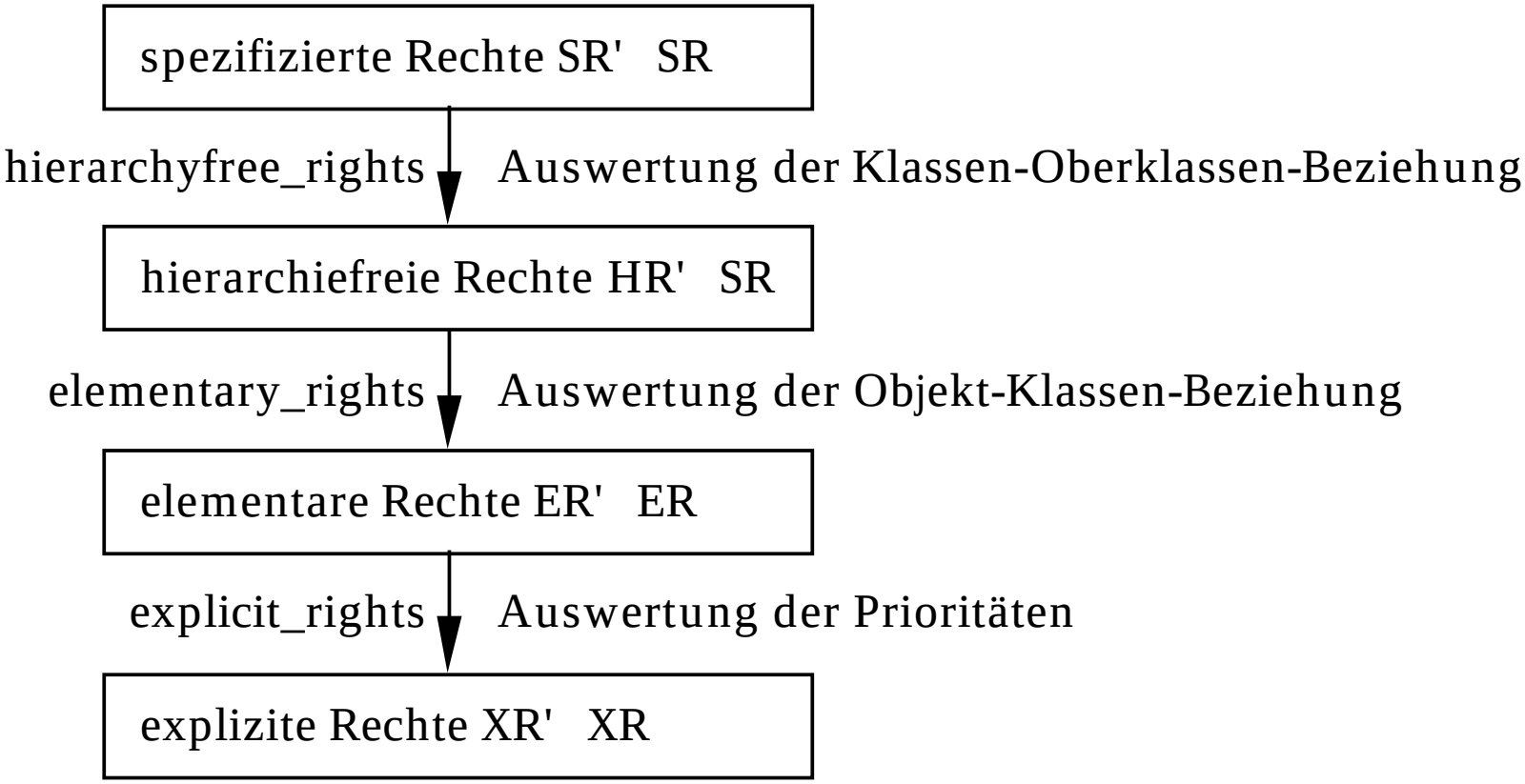
Die expliziten Rechte sind vor allem dafür gedacht, eine klare Semantik für die spezifizierten Rechte zu haben.

Da die Anzahl der expliziten Rechte i.a. erheblich größer ist als die Anzahl der spezifizierten Rechte, ist weder daran gedacht, die expliziten Rechte wirklich zu berechnen (außer vielleicht für Planspiele des Rechteverwalters) oder abzuspeichern, noch Anfragen des Zugriffskontrollsystems unmittelbar mit expliziten Rechten zu beantworten.

7.3.1 Der Erzeugungsprozess

Wir beschreiben den Erzeugungsprozess, bei dem aus einer Menge spezifizierter Rechte eine Menge expliziter Rechte erzeugt wird, in mehreren Schritten.

Wir erzeugen aus einer Menge spezifizierter Rechte zunächst hierarchiefreie Rechte, dann elementare Rechte und schließlich daraus explizite Rechte.



Prozess der Rechteerzeugung

Diese Schritte des Erzeugungsprozesses werden nachfolgend durch folgende Funktionen beschrieben:

Die Funktion hierarchyfree_rights wertet die Klassen-Oberklassen-Beziehung aus,
die Funktion elementary_rights die Objekt-Klassen-Beziehung und
die Funktion explicit_rights die Prioritäten.

Bei allen diesen Funktionen handelt es sich um idempotente Operationen.

7.3.2 Auswertung der Klassen-Oberklassen-Beziehungen

Wir benutzen hier -passend zu unserem Beispiel- für die Klassenhierarchien folgende Bedeutungen:

als Bedeutung der Subjektklassenhierarchie "**ist Untergebener von**" oder analog "**darf weniger**",
als Bedeutung der Operationsklassenhierarchie "**ist nicht so sensibel**" und
als Bedeutung der Granulklassenhierarchie "**ist Teil von**".

Andere Bedeutungen für die Klassenhierarchien sind jedoch möglich.

Um so nah wie möglich bei der intuitiven Semantik zu bleiben, haben wir die folgenden Strategien für die Behandlung der Hierarchien gewählt:

Bei Granulen sind die Hierarchien für Erlaubnisse und Verbote gleich;
bei Subjekten und Operationen ist die Hierarchie für Verbote invers zur der Hierarchie für Erlaubnisse.

Wir untersuchen damit beide theoretisch interessanten Fälle:
Erlaubnisse und Verbote sind gleichläufig und Erlaubnisse und Verbote sind gegenläufig.

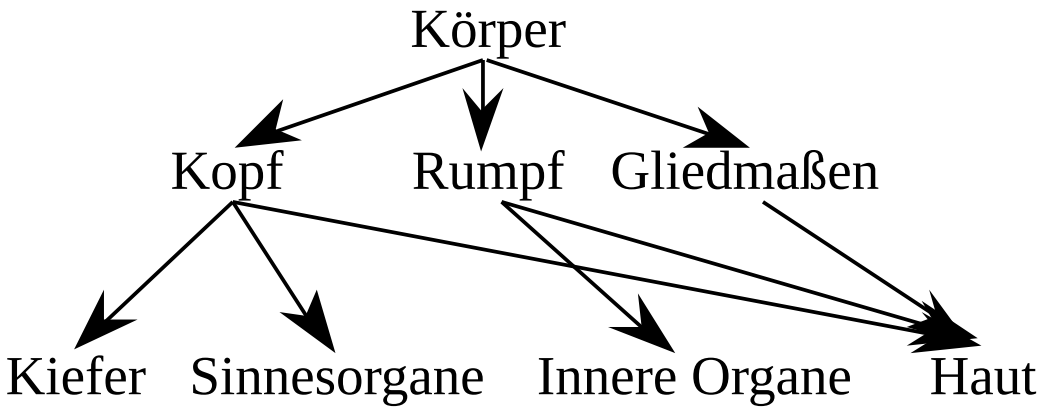
7.3.2.1 Gleichläufige Hierarchiebehandlung

Bei der gleichläufigen Hierarchiebehandlung wird für Erlaubnisse und Verbote die gleiche (nämlich stets die spezifizierte) Klassenhierarchie verwendet.

Wir behandeln hier die "ist Teil von"-**Klassenhierarchie für Granule** gleichläufig:

Ein Recht, das für eine Granulklasse spezifiziert wurde, ist gültig für alle ihre Unterklassen. Z.B. ist ein Recht (Erlaubnis oder Verbot) für die Klasse Kopf auch gültig für die Klasse Kiefer.

Für unser Beispiel ergibt sich daher als Klassenhierarchie für Verbote:



Hierarchie der Granulklassen (für Verbote)

Weitere mögliche Hierarchiebedeutungen sind z.B. "**is a**", "**is version of**"; sie können aber formal genauso gleichläufig wie "**is part of**" behandelt werden.

7.3.2.2 Gegenläufige Hierarchiebehandlung

Bei der gegenläufigen Hierarchiebehandlung werden für Erlaubnisse und Verbote unterschiedliche Klassenhierarchien verwendet, und zwar die spezifizierte Hierarchie für Erlaubnisse und die invertierte Hierarchie für Verbote.

Wir behandeln hier die "darf weniger"-**Klassenhierarchie für Subjekte** und die "ist nicht so sensibel"-**Klassenhierarchie für Operationen** gegenläufig:

Eine Erlaubnis, die für eine Subjektklasse spezifiziert wurde, ist gültig für alle ihre Unterklassen.
Z.B. ist eine Erlaubnis für die Klasse Krankenschwester auch gültig für die Klasse Arzt.

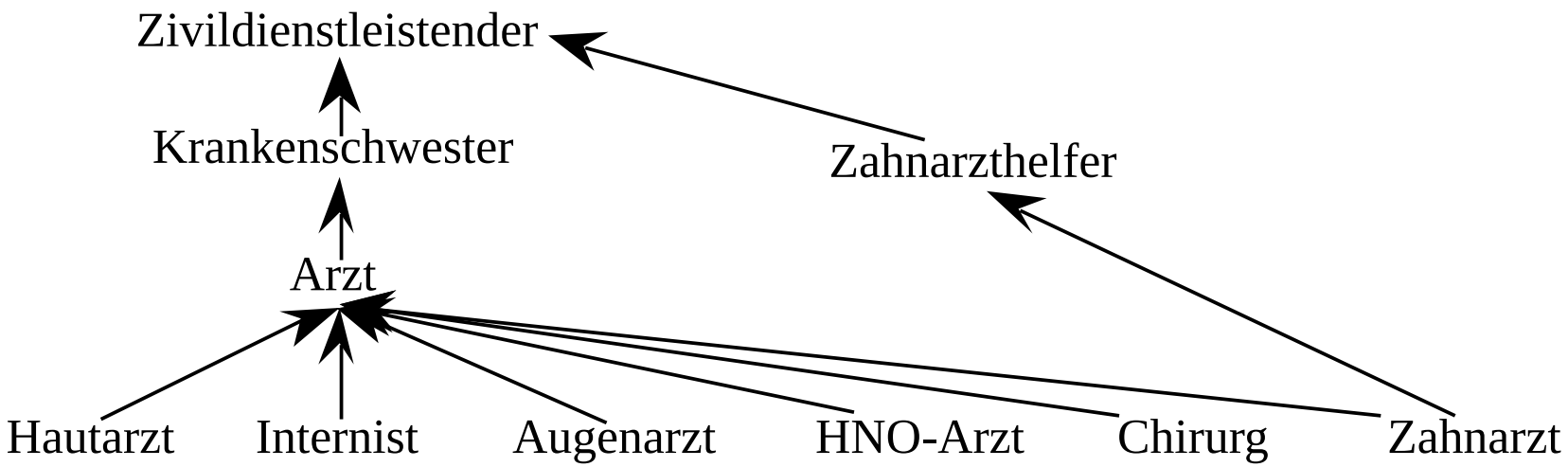
Ein Verbot, das für eine Subjektklasse spezifiziert wurde, ist gültig für alle ihre Oberklassen (bzgl. der Erlaubnishierarchie).
Z.B. ist ein Verbot für die Klasse Arzt auch gültig für die Klasse Krankenschwester.

Eine Erlaubnis, die für eine Operationsklasse spezifiziert wurde, ist gültig für alle ihre Unterklassen.
Z.B. ist eine Erlaubnis für die Klasse Med. Operation auch gültig für die Klasse Diagnose.

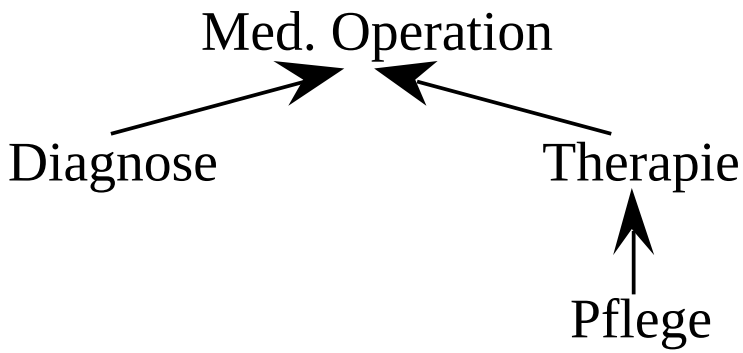
Ein Verbot, das für eine Operationsklasse spezifiziert wurde, ist gültig für alle ihre Oberklassen (bzgl. der Erlaubnishierarchie).
Z.B. ist ein Verbot für die Klasse Diagnose auch gültig für die Klasse Med. Operation.

Drehen wir also bei den "darf weniger"- und "ist nicht so sensibel"-Beziehungen für Verbote die Klassenhierarchie-Richtung um, so erhalten wir auch für Verbote, dass -wie gewohnt- Rechte für eine Klasse implizit auch in allen ihren Unterklassen gelten.

Für unser Beispiel ergeben sich daher als Klassenhierarchie für Verbote:



Hierarchie der Subjektklassen für Verbote



Hierarchie der Operationsklassen für Verbote

7.3.2.3 Formalisierung der Hierarchiebehandlung

Die Wirkung der Expansion der Klassenhierarchie ist formal beschrieben durch die Funktion subclasses, die die Menge der Unterklassen bzgl. der entsprechenden Klassenhierarchie berechnet.

Wir kennzeichnen jetzt die Klassenhierarchien außer durch die Kategorie auch durch die Rechtekennung.
Ist die zugrundegelegte Hierarchiebedeutung klar, so lassen wir den Index weg.

Beispiel:

$subclasses_{Untergebener\_von,\ Erlaubnis}(Arzt)$ = {Arzt, Hautarzt, Internist, Augenarzt, HNO-Arzt, Chirurg, Zahnarzt}
$subclasses_{Untergebener\_von,\ Verbot}(Arzt)$ = {Arzt, Krankenschwester, Zivildienstleistender}.

Unter theoretischen Gesichtpunkten unterscheiden wir nicht nach der Subjekt-, Operations- und Granulhierarchie, sondern zwischen gleich- und gegenläufigen Hierarchien.

Bemerkung: Sei c eine Klasse.
Für gleichläufige Hierarchien erhalten wir:

(a) $subclasses_{gleich,\ Verbot}(c) = subclasses_{gleich,\ Erlaubnis}(c)$
(b) $superclasses_{gleich,\ Verbot}(c) = superclasses_{gleich,\ Erlaubnis}(c)$

Für gegenläufige Hierarchien erhalten wir:

(c) $subclasses_{gegen,\ Verbot}(c) = superclasses_{gegen,\ Erlaubnis}(c)$
(d) $superclasses_{gegen,\ Verbot}(c) = subclasses_{gegen,\ Erlaubnis}(c)$

Bemerkung:
Wir lassen dabei den ersten Index weg, wenn die Eigenschaft sowohl für gegenläufige als auch für gleichläufige Hierarchien gilt und den zweiten Index, wenn die Eigenschaft für Erlaubnisse gilt.

In Übereinstimmung mit unseren bisherigen Bezeichnungen schreiben wir dann für die Erlaubnishierarchien kurz:
$subclasses(c) := subclasses_{gleich,\ Erlaubnis}(c) = subclasses_{gegen,\ Erlaubnis}(c)$.
$superclasses(c) := superclasses_{gleich,\ Erlaubnis}(c) = superclasses_{gegen,\ Erlaubnis}(c)$.

7.3.2.4 Hierarchiefreie Rechte

Wir ersetzen jetzt jede Klasse durch die von ihr überdeckten Klassen, und erhalten so **hierarchiefreie Rechte**.
Hierarchiefreie Rechte unterscheiden sich syntaktisch nicht von spezifizierten Rechten.
In hierarchiefreien Rechten steht jedoch jede Klasse nur für sich selbst.

Die Menge der zu einer Menge von spezifizierten Rechten SR' gehörigen hierarchiefreien Rechten ergibt sich, indem die Subjekt-, Operations- und Granulhierarchien unabhängig voneinander ausgewertet werden:

Die boolesche Funktion is_object prüft, ob ihr aktueller Parameter eine Klasse oder ein Objekt ist.

Definition: Sei ss S SC, so O OC, sg G GC, t TAG, p PRIO, sa (S SC) × (O OC) × (G GC), SR' SR.
covered_items$_{S,t}$(ss) := {hs S SC | if is_object(ss) then hs=ss else hs subclasses$_{S,t}$(ss) fi}
(und für Operationen und Granule analog)

hierarchyfree_actions(t,ss,so,sg) := {(hs,ho,hg) A | hs covered_items$_{S,t}$(ss)
ho covered_items$_{O,t}$(so)
hg covered_items$_{G,t}$(sg) },

hierarchyfree_rights(t,p,sa) := {(t,p,ha) SR | ha hierarchyfree_actions(t,sa)}

Aus einer Menge von spezifizierten Rechten SR' ergeben sich dann folgendermaßen die hierarchiefreien Rechte:
hierarchyfree_rights(SR') := $\bigcup_{sr \in SR'}$ hierarchyfree_rights(sr)

Beispiel: Sei sr=(Verbot, 20, Arzt, transplantieren, Körper). Dann ist
$covered\_items_{S,\ Verbot}$(Arzt) = {Arzt, Krankenschwester, Zivildienstleistender},
$covered\_items_{O,\ Verbot}$(transplantieren) = {transplantieren},
$covered\_items_{G,\ Verbot}$(Körper) =
{Körper, Kopf, Rumpf, Gliedmaßen, Kiefer, Sinnesorgane, Innere Organe, Haut}

und daher

hierarchyfree_rights(sr) = {
(Verbot, 20, Arzt, transplantieren, Körper),
(Verbot, 20, Krankenschwester, transplantieren, Körper),
(Verbot, 20, Zivildienstleistender, transplantieren, Körper),
(Verbot, 20, Arzt, transplantieren, Kopf),
(Verbot, 20, Krankenschwester, transplantieren, Kopf),
(Verbot, 20, Zivildienstleistender, transplantieren, Kopf),
(Verbot, 20, Arzt, transplantieren, Rumpf),
(Verbot, 20, Krankenschwester, transplantieren, Rumpf),
(Verbot, 20, Zivildienstleistender, transplantieren, Rumpf),
(Verbot, 20, Arzt, transplantieren, Gliedmaßen),
(Verbot, 20, Krankenschwester, transplantieren, Gliedmaßen),
(Verbot, 20, Zivildienstleistender, transplantieren, Gliedmaßen),
(Verbot, 20, Arzt, transplantieren, Kiefer),
(Verbot, 20, Krankenschwester, transplantieren, Kiefer),
(Verbot, 20, Zivildienstleistender, transplantieren, Kiefer),

(Verbot, 20, Arzt, transplantieren, Sinnesorgane),
(Verbot, 20, Krankenschwester, transplantieren, Sinnesorgane),
(Verbot, 20, Zivildienstleistender, transplantieren, Sinnesorgane),
(Verbot, 20, Arzt, transplantieren, Innere Organe),
(Verbot, 20, Krankenschwester, transplantieren, Innere Organe),
(Verbot, 20, Zivildienstleistender, transplantieren, Innere Organe),
(Verbot, 20, Arzt, transplantieren, Haut),
(Verbot, 20, Krankenschwester, transplantieren, Haut),
(Verbot, 20, Zivildienstleistender, transplantieren, Haut) }

Schon hier erkennt man die stark abkürzende Funktion der Hierarchien.

7.3.3 Auswertung der Objekt-Klassen-Beziehungen

In diesem Schritt werden die Klassennamen durch die Menge der Objektnamen ihrer Klassenmitglieder ersetzt und das Ergebnis elementweise ausgewertet. Genauso kann man Objektmengen auswerten.

7.3.3.1 Elementare Rechte

Durch die Ersetzung erhalten wir eine Menge von Rechten ohne Klassennamen, sog. elementare Rechte.

Definition: Die Menge aller elementaren Rechte ist ER:=TAG×PRIO×EA.
Ein **elementares Recht** ist also ein Tupel er=(t,p,ea) TAG×PRIO×EA.
Dabei enthält die Handlungskomponente jetzt ausschließlich Objekte (also keine Klassen).

Definition: Sei hs S SC, ho O OC, hg G GC. Die boolesche Funktion is_object prüft, ob ihr aktueller Parameter eine Klasse oder ein Objekt ist.
covered_objects(hs) := if is_object(hs) then {hs} else mc_S(hs) fi
covered_objects(ho) := if is_object(ho) then {ho} else mc_O(ho) fi
covered_objects(hg) := if is_object(hg) then {hg} else mc_G(hg) fi

Beispiel: covered_objects(Arzt) = mc_S(Arzt) = {john, jane}

Subjekte, Operationen und Granule werden unabhängig voneinander ausgewertet:

Definition: Sei hr=(t,p,ha) ein hierarchiefreies Recht mit
ha=(hs,ho,hg) (S SC) × (O OC) × (G GC), t TAG, p PRIO, HR' SR.

elementary_actions(hs,ho,hg) := {(es,eo,eg) EA | es covered_objects(hs)
eo covered_objects(ho)
eg covered_objects(hg) }

elementary_rights(t,p,ha) := {(t,p,ea) ER | ea elementary_actions(ha)}

elementary_rights(HR') := $_{hr \in HR'}$ elementary_rights(hr)

ist die Menge aller elementaren Rechte, die aus einer Menge von hierarchiefreien Rechten HR' abgeleitet werden können.

7.3.4 Auswertung der Prioritäten

Wir berechnen die Menge der expliziten Rechte aus der Menge aller elementaren Rechte, indem wir für jede elementare Handlung ea das Recht mit der maximalen Priorität auswählen.

(Falls es mehrere solche Rechte gibt, die verschieden sind, liegt ein Konflikt vor).

Die Rechtekennung dieses Rechts zeigt uns, ob eine explizite Erlaubnis oder ein explizites Verbot vorliegt.

Definition: Sei ER' ER.
maximal_priority(ER') := {(t,p,ea) ER' | er' ER': er'=(t',p',ea) ¬(p'>p)} [7]

forget_priority(ER') := {(t,ea) XR | p PRIO: (t,p,ea) ER'}

Mit diesen Definitionen können wir beschreiben, welche expliziten Rechte aus einer Menge elementarer Rechte abgeleitet werden können:

explicit_rights(ER') := forget_priority(maximal_priority(ER'))

Beispiel: Nun können wir die spezifizierten Rechte auswerten und anhand der ermittelten expliziten Rechte z.B. überprüfen, dass nur Hendrik und Anne Lungentransplantationen ausführen dürfen, d.h. dass insbesondere

(Erlaubnis, hendrik, transplantieren, lunge),
(Erlaubnis, anne, transplantieren, lunge) explicit_rights(elementary_rights(hierachyfree_rights(SR_1)))

gilt.

---

[7] Für lineare Prioritätsordnungen würde die Bedingung p'≤p statt ¬(p'>p) ausreichen. Wir behandeln hier den Fall schon mit, dass Prioritäten unvergleichbar sein können.