Hierarchisches Planen durch Propositionale Logik – Hohe Flexibilität, Vielseitigkeit und Flexibilität[1]

Gregor Behnke[2]

Abstract: Planungsbasierte Assistenzsysteme bieten ihren Nutzern flexible und individualisierte Unterstützung bei der Bewältigung schwieriger Probleme. Für einen erfolgreichen Assistenten wird ein effizientes und flexibles Planungssystem benötigt. Wir demonstrieren, wie mittels einer Transformation in Aussagenlogik der momentan schnellste HTN-Planer konstruiert werden kann und veranschaulichen, auf welche Weise dieser garantiert optimale Lösungen finden kann. Weiterhin zeigen wir, dass die Wünsche des Nutzers in Linearer Temporaler Logik ausgedrückt werden können – und wie dieser in den Planungsprozess einbezogen werden kann. Letztlich stellen wir dar, wie die entwickelten Technologien in Kooperation mit der Robert Bosch GmbH in einem industriellen Kontext verwendet werden. Der im Rahmen dieser Zusammenarbeit entwickelte Assistent ROBERT hilft seinen Nutzern bei der Durchführung von Do-It-Yourself Heimwerker-Projekten.

1 Einführung

Planungstechniken der Künstlichen Intelligenz ermöglichen es, individualisierte und situations-adaptive Unterstützung für vielfältige Problemstellungen anzubieten. Planungsbasierte Assistenzsysteme geben individuelle und kontextabhängige Anweisungen zur Lösung von komplexen Problemen. Sie können z.B. bei technischen Problemen, Haushaltsreparaturen, Do-It-Yourself-Projekten oder schwierigen Montageaufgaben Hilfestellung leisten. Als Basis für diese Assistenz dienen die von Planungssystemen generierten Handlungspläne, die Sequenzen von einzelnen Aktionen sind. Wenn diese ausgeführt werden, lösen sie die vom Nutzer gestellte Aufgabe. Der Handlungsplan dient als Grundlage für eine Schritt-für-Schritt Anleitung, die dem Nutzer präsentiert wird. Der Plan wird individuell für die konkrete Situation, in der sich der Nutzer befindet (z.B. vorhandene Werkzeuge und Materialien), generiert. Der Planer passt die dem Nutzer gegebenen Anweisungen perfekt an die aktuelle Situation an. So verhindert er, dass die Aufgabe etwa wegen fehlenden Materials oder eines fehlenden Werkzeugs scheitert. Planungsbasierte Assistenz sollte den aktuellen Wünschen und Präferenzen des Benutzers Rechnung tragen. Der Assistent muss verhindern, dass er dem Nutzer Instruktionen gibt, die dieser nicht wünscht oder für unangebracht hält. Dies wird durch die Einbeziehung des Nutzers in den Planungsprozess erreicht – durch *gemischt-initiatives* Planen [Sm12]. Im Rahmen dieser Dissertation wurden Planungsprobleme betrachtet, die mittels hierarchischer Task Netzwerke (HTN, [EHN96]) modelliert wurden. Für die Unterstützung (menschlicher) Nutzer eignen sich HTNs besonders gut, da ihre Art Probleme zu beschreiben strukturell der menschlichen Problemlösestrategie ähnelt [By77, WM05].

[1] Englischer Titel der Dissertation: „Hierarchical Planning through Propositional Logic – Highly Efficient, Versatile, and Flexible“

[2] Albert-Ludwigs-Universität Freiburg, Institut für Informatik, behnkeg@informatik.uni-freiburg.de

Das Hauptaugenmerk dieser Arbeit [Be19a] liegt auf der Entwicklung eines effizienten HTN-Planungssystems, das in der Lage ist, Pläne ausreichend schnell zu finden (Kapitel 3). Hierzu wurde ein neuer Lösungsansatz verwendet: die Transformation des Planungsproblems in Aussagenlogische Formeln. Aus erfüllenden Belegungen dieser Formeln wird anschließend der Plan extrahiert. Die Transformation basiert auf neuartigen Datenstrukturen (Path Decomposition Graphs und Solution Order Graphs), die es erlauben, die Struktur eines HTN-Planungsproblems kompakt darzustellen. Außerdem wurde gezeigt, wie die entwickelten Technologien verwendet werden können, um garantiert *optimale* Pläne zu finden. Dies ist insbesondere für Assistenzsysteme wichtig, da nicht-optimale Pläne zu Anweisungen führen können, die einem Nutzer unsinnig erscheinen. Dies führt dazu, dass der Assistent in Zukunft nicht mehr verwendet wird.

Um Anfragen und Anforderungen des Nutzers in den Planungsprozess einbinden zu können, müssen diese zunächst geeignet repräsentiert werden. Hierzu verwenden wir Formeln in Linearer Temporaler Logik (LTL). Wir untersuchen sowohl die theoretischen als auch die praktischen Aspekte des HTN-Planens unter gegebenen LTL-Formeln (Kapitel 4) .

In einem DFG-geförderten Transferprojekt mit der Robert Bosch GmbH wurde gezeigt, dass die entwickelten Techniken auch in einem industriellen Kontext erfolgreich eingesetzt werden können (Kapitel 5). Es wurde das Assistenzsystem ROBERT entwickelt, das Nutzer durch ein handwerkliches Do-It-Yourself-Projekt führt. Mit ROBERT können Nutzer selbst komplexeste Projekte erfolgreich beenden.

2 Hierarchisches Planen

Beim Planen mit hierarchischen Task-Netzwerken (HTN, [EHN96]) werden zwei Arten von Aktivitäten unterschieden: primitive Aktionen und abstrakte Tasks. Während primitive Aktionen über eine Zustands-Übergangs-Semantik definiert sind, repräsentieren abstrakte Tasks komplexere Handlungsfolgen. Ihre Semantik ist über sogenannte Dekompositionsmethoden definiert. Eine solche Methode erlaubt die Ausführung eines abstrakten Task t, indem statt t eine partiell geordnete Menge tn von anderen Tasks und Aktionen ausgeführt wird. Diese Menge tn nennt man *Task Netzwerk*. Das Ziel eines HTN-Planungsproblems ist durch einen abstrakten Task gegeben, der ausgeführt werden soll. Dies könnte z.B. `baue-ein-Vogelhaus` sein. Um einen Plan zu erhalten, muss nun eine Dekompositionsmethode auf den initialen Task angewendet werden. Das Resultat ist ein Task Netzwerk, das wiederum abstrakte Tasks enthalten kann. Auch auf diese müssen Dekompositionsmethoden angewendet werden. Dieser Prozess wird wiederholt, bis ein Task-Netzwerk gefunden wird, das nur primitive Aktionen enthält und dessen Aktionen im gegebenen initialen Zustand in der gegebenen Reihenfolge ausführbar sind. Folglich ist ein Plan genau dann eine Lösung für ein HTN-Planungsproblem, wenn er ausführbar ist und es eine Folge von Anwendungen von Dekompositionsmethoden – eine *Ableitung* – gibt, die ihn aus dem initialen abstrakten Task entstehen lässt.

3 Hierarchisches Planen mittels Aussagenlogik

Die Entwicklung effizienter Planer für HTN-Planungsprobleme ist besonders schwierig. Algorithmen müssen beide Dimensionen der Problemstellung – Zustandssemantik der Aktionen und Dekompositionsmethoden – und ihre Wechselwirkungen berücksichtigen. Traditionellerweise werden HTN Planungsprobleme durch Suche – im Plan- oder Progressionsraum – gelöst. Wir präsentieren eine neue Möglichkeit zur Lösung von HTN-Planungsproblemen: die Transformation in Aussagenlogische Formeln. Dabei wird eine Formel $\mathscr{F}$ gesucht, die genau dann erfüllbar ist, wenn das gegebene HTN-Planungsproblem eine Lösung besitzt. Eine erfüllende Belegung kann mit Hilfe eines der existierenden, effizienten SAT Solver gefunden werden. Aus dieser Belegung kann anschließend ein Plan extrahiert werden. Dieser kann die Grundlage für eine Schritt-für-Schritt Anleitung sein.

Da HTN Planung im Allgemeinen unentscheidbar ist [EHN96, GB11], kann es eine solche Transformation im Allgemeinen nicht geben. Stattdessen muss für ein individuelles Planungsproblem eine (potentiell unendliche) Sequenz $(\mathscr{F}_k)$ von Aussagenlogischen Formeln angegeben werden. Sofern das Planungsproblem eine Lösung hat, wird eine dieser Formeln erfüllbar sein – ebenso wie alle folgenden. Wir können folglich die $\mathscr{F}_k$ sequentiell auf Erfüllbarkeit testen.

Für die k-te generierte Formel betrachten wir eine Beschränkung des HTN-Planungsproblems: die k-Tiefenbeschränkung. Wir suchen mittels der k-te Formel also nur diejenigen Pläne, die über eine Zerlegung von höchstens der Tiefe k aus dem initialen abstrakten Task abgeleitet werden können. Das heißt, es ist eine Formel $\mathscr{F}_k$ zu konstruieren, die genau dann erfüllbar ist, wenn ein solcher Plan existiert. Diese Formel muss die möglichen Pläne und deren Ableitungen vom initialen abstrakten Task repräsentieren. Eine explizite Beschreibung all dieser möglichen Ableitungen ist nicht möglich, da es bis zu $\mathscr{O}(2^{2^k})$ solcher Ableitungen geben kann. Folglich benötigen wir eine kompakte Repräsentation der möglichen Pläne und ihrer Ableitungen.

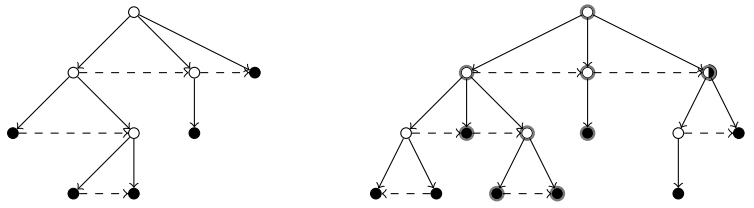

Abb. 1: Eine mögliche Ableitung eines HTN Planes (links) und ein Path Decomposition Tree (PDT, rechts). Weiße Knoten beschreiben abstrakte Tasks, schwarze primitive Aktionen. Der PDT enthält die links gezeigte Ableitung als Teilbaum. Die entsprechenden Knoten sind rot eingefärbt.

Hierzu führen wir das Konzept der Path Decomposition Trees (PDTs) ein. Der k-PDT eines HTN-Planungsproblems ist ein Baum, der alle möglichen k-tiefenbeschränkten Ableitungen als Teilbäume enthält. Abbildung 1 zeigt eine Ableitung und einen PDT, der die Ableitung als Teilbaum enthält. Eine erfüllende Belegung der zu konstruierenden Formel $\mathscr{F}_k$ wird nun einen solchen Teilbaum auswählen. Hierfür hat es für jeden Knoten v des PDT und Task t eine Variable t_v. Falls für einen Knoten v eine der Variablen t_v wahr ist,

repräsentiert dies, dass der Knoten Teil der ausgewählten Ableitung ist. Die einzelnen t_v kodieren, welcher Task an der entsprechenden Stelle in der Ableitung vorkommt.

Neben der Wahl einer konkreten Ableitung über die Belegung der a_v Variablen muss weiterhin sichergestellt werden, dass die resultierenden primitiven Aktionen im initialen Zustand ausführbar sind. Hierfür können existierende Kodierungen aus dem klassischen Planen [RHN06] verwendet werden [BHB19a, BHB18a, BHB18b]. Die abgeleiteten primitiven Aktionen müssen weiterhin in einer mit der Ableitung kompatiblen Reihenfolge ausführbar sein. Hierzu wurden drei verschiedene Möglichkeiten der Kodierung in Aussagenlogik entwickelt. Zunächst haben wir eine Spezialkodierung für den Fall entwickelt, dass alle Dekompositionsmethoden Sequenzen von Tasks und Aktionen sind [BHB18a]. Die zweite, allgemeine Kodierung repräsentiert die durch die Ableitung implizierten Ordnungsbeschränkungen mithilfe zusätzlicher Variablen [BHB18b]. Im Allgemeinen gibt es keine eindeutige Ordnung zwischen den Blättern des PDT – welche die primitiven Aktionen repräsentieren, die aus der Ableitung entstehen. Die resultierende Kodierung in Aussagenlogik hat $\mathcal{O}(n^4)$ Klauseln, wobei n die Anzahl der Blätter des PDT ist.

Wir haben gezeigt, dass dies signifikant verbessert werden kann. Hierzu schlagen wir vor, gemeinsam mit dem PDT eine partielle Ordnung seiner Blätter zu bestimmen. Diese wird durch einen gerichteten azyklischen Graphen, den Solution Order Graph (SOG), repräsentiert [BHB19a]. Die Knoten des SOG sind die Blätter die PDT, also die möglicherweise in einem abgeleiteten Plan enthaltenen primitiven Aktionen. Der SOG ist dann derart zu bestimmen, dass jedes konkrete ableitbare Task Netz ein induzierter Teilgraph des SOG ist. Hierdurch wird garantiert, dass, falls ein Blatt des PDT Teil der konkret gewählten Ableitung ist, dessen relative Ordnung zu anderen ebenfalls aus der Ableitung entstandenen Tasks immer die selbe, also statisch ist. Dies erlaubt eine erheblich kompaktere Kodierung mit lediglich $\mathcal{O}(n^2\Delta)$ vielen Klauseln, wobei Δ der maximale Ein-Grad eines Knotens im SOG ist. Neben dieser Anwendung zur Erstellung einer kompakteren aussagenlogischen Kodierung ist die Struktur des SOG auch vielversprechend für zukünftige Untersuchungen. Da er *alle* (k-tiefenbeschränkten) Lösungen eines HTN-Planungsproblems kompakt repräsentiert, kann er die Basis für die Extraktion von Eigenschaften des Planungsproblems, z.B. von HTN-spezifischen Mutexen o.ä. sein.

Fraglich ist nun, wie ein solcher PDT und der zugehörige SOG effizient berechnet werden können. Es ist praktisch unmöglich, alle Ableitungen zu berechnen und anschießend einen passenden PDT zu bestimmen. Stattdessen verwenden wir einen Algorithmus, der auf lokaler Expansion basiert. Wir beginnen mit einem PDT, der lediglich einen Wurzelknoten w enthält und weisen diesem eine Menge $\alpha(w) = \{a_I\}$ von möglichen Tasks und Aktionen zu, die für w genau den initialen abstrakten Task enthält. Nun betrachten wir wiederholt ein Blatt b des aktuellen PDTs und dessen Task-Menge $\alpha(b)$. Wir bestimmen dann alle auf die abstrakten Tasks in $\alpha(b)$ anwendbaren Methoden und deren Tasknetze. Wir interpretieren diese als eine Familie von Graphen (G_i). Anschließend bestimmen wir einen Graphen G^*, so dass alle G_i induzierte Teilgraphen von G^* sind. Wie wir gezeigt haben, ist bereits die Bestimmung von G^* $\mathbb{NP}$-vollständig[3] [BHB19a], jedoch nötig um einen SOG aus dem PDT zu extrahieren. Wir verwenden deshalb einen Greedy-Algorithmus, um G^* im Planer

[3] Das Entscheidungsproblem fragt, ob ein G^* mit weniger als x Knoten existiert.

zu bestimmen. Wir fügen nun die Knoten von G^* als Kinder von b in den PDT ein und bestimmen die Menge α für jeden Knoten in G^*, basierend auf den Homomorphismen der G_i in G^*. Wir wiederholen diesen Prozess, bis alle Blätter des PDT die Tiefe k haben.

Der beschriebene Prozess der Erstellung eines PDT/SOG-Paares wurde automatisiert und auf seiner Basis ein SAT-basierter HTN-Planer auch praktisch umgesetzt. Dieser Planer wurde auf einer Menge von Benchmark-Instanzen gegen andere State-of-the-Art HTN-Planungssysteme verglichen[4]. Die jeweils pro Domäne und insgesamt gelösten Instanzen sind in Tabelle 1 dargestellt. Es lässt sich erkennen, dass der neu entwickelte SAT-basierte HTN-Planer signifikant mehr Instanzen als bisherige Techniken löst. Insbesondere schwerere und komplexere Planungsprobleme lassen sich mit dem neuen Ansatz leichter und deutlich schneller lösen.

	#instances	SAT-F ∃ expMC	SAT-F ∃ MapleLCM	SAT-F ∃ CaDiCaL	SAT-F ∃ cryptominisat	SAT-F expMC	SAT-F MapleLCM	SAT-F CaDiCaL	SAT-F cryptominisat	SAT-tree ∃ expMC	SAT-tree ∃ MapleLCM	SAT-tree ∃ CaDiCaL	SAT-tree ∃ cryptominisat	SAT-tree expMC	SAT-tree MapleLCM	SAT-tree CaDiCaL	SAT-tree cryptominisat	PANDApro lm-cut	PANDApro FF	PANDApro ADD	TDG-m greedy A*	TDG-c greedy A*	UMCP H	UMCP BF	UMCP DF	HTN2STRIPS jasper	HTN2STRIPS FD-SS 2018	HTN2STRIPS SaarPlan	HTN2STRIPS LAPKT-BFWS	HTN2STRIPS MpC	SHOP2	FAPE
UM-TRANSLOG	22	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	**22**	19	17	17	17	6	**22**	-
SATELLITE	25	**25**	**25**	**25**	**25**	24	24	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	24	23	**25**	21	23	18	20	23	19	14	12	0	22	22
WOODWORKING	11	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	**11**	10	9	9	8	10	6	6	6	5	5	5	5	4	8	0
SMARTPHONE	7	**7**	**7**	**7**	**7**	**7**	**7**	6	**7**	6	6	**7**	**7**	6	6	6	**7**	5	5	5	5	5	4	4	4	6	6	5	5	4	4	-
PCP	17	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	11	**12**	11	**12**	11	**12**	9	10	11	9	8	0	0	0	3	3	3	3	0	0	-
ENTERTAINMENT	12	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	11	11	**12**	9	9	6	5	5	5	5	5	4	4	5	-
ROVER	20	10	**11**	9	8	5	6	4	4	4	4	4	6	4	4	4	5	4	3	4	2	2	0	0	0	5	5	4	4	4	3	3
TRANSPORT	30	**22**	20	20	20	15	14	15	17	**22**	20	19	21	15	15	15	18	9	11	7	1	1	0	1	0	19	17	13	13	3	0	-
total	144	**121**	120	118	117	108	108	107	110	114	112	111	116	106	107	106	112	95	95	93	81	78	61	56	57	85	77	66	63	25	64	25/56

Tab. 1: Anzahl der von verschiedenen HTN-Planungssystemen gelösten Benchmark-Instanzen [BHB19a]. SAT-tree beschreibt die Kodierung mittels zusätzlicher Variablen. SAT-F beschreibt die Kodierung auf Basis des SOG. Kodierungen mit ∃ verwenden die klassische ∃-step Kodierung [RHN06], diejenigen ohne die Kautz&Selman-Kodierung [KS96]. Zuletzt wird der jeweils verwendete SAT-Solver genannt. PANDApro [Hö19, Hö18], TDG [Be17], UMCP [EHN94], HTN2STRIPS [Al16], SHOP2 [Na03] und FAPE [Dv14] sind HTN-Planer aus der Literatur.

Die bis hierhin beschriebene Technik zum Finden von Plänen mittels einer Transformation in Aussagenlogik ermöglicht nur das Finden *irgendeines* Planes. Insbesondere in planungsbasierten Assistenzsystemen ist es jedoch von großer Bedeutung, einen Kosten-optimalen Plan zu finden. Nicht-optimale Pläne enthalten regelmäßig überflüssige Aktionen. Womöglich könnte ein solcher Plan den Nutzer anweisen, eine Batterie aus einem Gerät zu entfernen und sofort wieder einzusetzen. Derartige Anweisungen sind für den Nutzer klar als unsinnig zu erkennen. Dies führt zu einem Verlust an Vertrauen in die Fähigkeiten des Assistenten und kann einen Abbruch der Interaktion mit ihm verursachen.

Mittels der entwickelten aussagenlogischen Kodierung kann auch ein Kosten-optimaler Plan gefunden werden [BHB19b]. Hierzu wird zunächst ein beliebiger Plan gesucht. Dessen Kosten C^* bilden eine obere Schranke für die Kosten des optimalen Plans. Anschließend wird der optimale Plan mittels weiterer SAT-Aufrufe unter einer Kosten-Beschränkung und binärer Suche gefunden. Für eine gegebene Kosten-Schranke C ist hierzu eine Tiefen-

[4] Dabei erhielt jeder Planer für jede Instanz 10 Minuten Zeit und 4 GB RAM auf einem Intel Xeon E5-2660.

beschränkung $k(C)$ derart zu wählen, dass jeder Plan mit höchstens Kosten C eine Tiefe von höchstens $k(C)$ besitzt. Wir haben gezeigt, wie eine solche Tiefenbeschränkung approximiert werden kann [BHB19b]. Experimente haben erwiesen, dass diese Approximation in der Regel sehr genau ist und dass der resultierende, optimale HTN-Planer ebenfalls andere Ansätze zum Finden optimaler HTN-Pläne signifikant dominiert. Wir haben damit insgesamt die Grundlage für die Verwendung von effizienten HTN-Planungssystemen für planungsbasierte Assistenzsysteme gelegt.

4 Änderung von Plänen

Planungsbasierte Assistenzsysteme müssen sowohl einen schnellen, effizienten Planer als auch einen flexiblen Planer besitzen. Dieser muss in der Lage sein, auf die Wünsche und Präferenzen des Nutzers einzugehen. Im Rahmen eines planungsbasierten Assistenzsystems wird der Nutzer diese bezüglich eines konkreten, vom Assistenten vorgeschlagenen Plans äußern, d.h. seine Wünsche werden die Gestalt einer Anfrage zur Änderung eines Planes haben. Wir haben zunächst die theoretischen Grundlagen derartiger Wünsche untersucht. Hier stellt sich zunächst die Frage, wie derartige Wünsche geeignet formal repräsentiert werden können. Wir schlagen vor, dafür Formeln in Linearer Temporaler Logik (LTL[5] [Pn77]) zu verwenden. So kann beispielsweise in einem Heimwerkerszenario der Wunsch „Ich möchte alle Sägekanten schleifen.“ als $\mathrm{G}(\mathit{sägen}(x) \rightarrow \mathrm{E}\mathit{schleifen}(x))$ modelliert werden. Derartige Formeln können mittels Techniken des maschinellen Lernens aus natürlich-sprachlichen Aussagen des Nutzers extrahiert werden.

Der Ausgangspunkt für unsere theoretischen Untersuchungen war das *Planverifikationsproblem*. Dieses stellt die Frage, ob eine gegebene Sequenz von primitiven Aktionen eine Lösung für ein gegebenes HTN-Planungsproblem darstellt. Es handelt sich dabei gleichzeitig um den speziellsten Wunsch, den ein Nutzer ausdrücken kann (eine Formel der Form $a_1 \wedge X(a_2 \wedge X(\ldots))$). Wir haben gezeigt, dass das Planverifikationsproblem $\mathbb{NP}$-vollständig ist [BHB15]. Basierend auf diesem Ergebnis haben wir die Berechnungskomplexität der elementarsten Anfragen zur Änderung eines Plans (Hinzufügen, Entfernen, Austauschen, Neuordnen von Aktionen) untersucht und gezeigt, dass diese dieselbe Komplexität wie das jeweilige Planexistenzproblem haben [Be16]. In der Dissertation wurde dieses Ergebnis auf allgemeine LTL-Formeln erweitert. Nach unserem besten Wissen – ist dies die erste komplexitätstheoretische Analyse des gemischten-initiativen HTN-Planens.

Um Wünsche und Anfragen des Nutzers nun auch praktisch beantworten zu können, haben wir entsprechende Techniken entwickelt. Zunächst haben wir einen Planverifizierer für HTN-Pläne erarbeitet, der auf einer Transformation des Problems in Aussagenlogik basiert [BHB17]. Er ist der erste Verifikator für Lösungen von HTN-Planungsproblemen. Planverifikation ist neben der Anwendung in planungsbasierten Assistenzsystemen auch für die Forschung von hoher Bedeutung, da sie die *unabhängige* Überprüfung von entwickelten Planungssystemen ermöglicht und so zur Objektivierung der Forschung beiträgt. Die Publikation unseres Planverifizierers hat bereits weitere Forschung anderer Gruppen angestoßen hat [BMC18].

[5] Technisch betrachten wir LTL über finiten Sequenzen, also LTLf [DGV13].

Der Planer muss nun in der Lage sein, einen Plan zu finden, der sowohl eine Lösung des eigentlichen Planungsproblems ist, als auch eine LTL Formel ϕ erfüllt. Als Basis verwenden wir den im vorherigen Kapitel beschriebenen HTN Planer auf Basis von Aussagenlogik. Der deklarative Charakter der Kodierung – sie modelliert schlicht, *was* eine Lösung ist – erlaubt es, Kodierungen für weitere Einschränkungen, wie z.B. LTL Formeln, hinzuzufügen. Da sich die LTL-Formeln (momentan) nur auf die primitiven Aktionen eines Planes beziehen, sind die für das klassische Planen entwickelten Kodierungen für LTL-Formeln auch auf HTN-Planungsprobleme anwendbar. Wir verwenden die Kodierung von Mattmüller und Rintanen [MR07]. Leider unterstützt diese Kodierung nicht alle syntaktisch gültigen LTL-Formeln, sondern nur solche ohne den Next-Operator X. Dieser Operator ist schwierig zu handhaben, da sein Vorhandensein die Ausnutzung der Stotteräquivalenz [La83] verbietet. Er ist jedoch in der Praxis notwendig, da er das einzige Mittel ist, um festzulegen, dass eine bestimmte Aufgabe ausgeführt werden muss, wenn die unmittelbar nächste Aktion – z.B. in einer Benutzeranforderung "Ich will als nächstes sägen" – ausgeführt wird. Wir konnten zeigen, dass der X-Operator trotz der fehlenden Stotteräquivalenz effizient in die Kodierung integriert werden kann [BB18]. Die hierfür notwendigen Änderungen führten zu weiteren Modifikationen an der bisherigen Kodierung, die diese ebenfalls effizienter machten.

Abb. 2: ROBERT erklärt deinem Nutzer wie er einen 3mm Metallbohrer erkennt und diesen in die Bohrmaschine einsetzt.

5 Planungsbasierte Assistenz in der Praxis

Im Rahmen der Dissertation wurden theoretische und technische Konzepte erarbeitet und umgesetzt. Darüber hinaus wurden diese in der praktischen, industrienahen Anwendung umgesetzt und evaluiert. Im Rahmen eines DFG-Transferprojekts gemeinsam mit der Robert Bosch GmbH wurde der planungsbasierte Assistent ROBERT entwickelt [Be19b, Be18]. Er unterstützt bei der Durchführung von Do-It-Yourself Heimwerkerprojekten. Ohne geeignete Assistenz überfordern diese schnell durch komplexe Anforderungen hinsichtlich angemessener Vorgehensweisen, Wahl der passenden Werkzeuge und Arbeitsmittel. Der multimediale und situations-adaptive Heimwerkerassistent ROBERT leitet seinen Nutzer

schrittweise durch das Heimwerkerprojekt. Die gegebene Anleitung wird durch den in dieser Dissertation entwickelten Planer bestimmt. Das dabei verwendete Planungsmodell ist sehr komplex, da es die verschiedensten Möglichkeiten zur Verwendung von Materialien und Werkzeugen beschreiben muss. Aktuell ist nur der hier präsentierte HTN-Planer in der Lage, die von ROBERT benötigten Pläne in einem akzeptablen Zeitrahmen zu finden.

ROBERT präsentiert die gefundenen Handlungsschritte als eine multimediale Anleitung, deren Inhalte dynamisch anhand der durchzuführenden Handlung bestimmt werden. Ein Beispiel für eine solche Anleitung wird in Abbildung 2 gezeigt. ROBERT ist auch in der Lage, Fragen des Nutzers mittels seines Hintergrundwissens passend zu beantworten. Insgesamt ermöglicht ROBERT so eine vollkommen neuartige Unterstützung – vor allem von bisher unerfahrenen Heimwerkern, denen bis jetzt nur textuelle Anleitungen und nicht individualisierte YouTube-Videos zur Verfügung standen. Durch seine detaillierte und vertrauenswürdige Assistenz ermöglicht ROBERT es Personen sogar, ihre „Angst“ vor elektrischen Heimwerkergeräten, wie z.B. einer Stichsäge, abzulegen und diese erstmals erfolgreich zu verwenden. ROBERT wurde in gemeinsam mit der Robert Bosch GmbH durchgeführten Studien mehrfach erfolgreich mit Probanden getestet [Be20, Be19b].

6 Zusammenfassung

Damit die Planer die Grundlage für eine sinnvolle und individualisierte Assistenz bieten können, müssen sie sowohl schnell als auch mit der Fähigkeit ausgestattet sein, Pläne entsprechend der Wünschen ihrer Nutzer zu ändern. Wir haben gezeigt, wie diese beiden Herausforderungen für hierarchische Planungsprobleme angegangen werden können. Zunächst wurde ein höchst effizienter neuer Ansatz zur Generierung von HTN-Plänen erarbeitet: HTN-Planen mittels Aussagenlogik. Hierzu wurden Path Decomposition Trees (PDTs) und Solution Order Graphs (SOGs) entwickelt, die eine kompakte Repräsentation eines k-tiefenbeschränkten HTN-Planungsproblems ermöglichen. Auf Basis von PDT und SOG entwickelten wir drei Kodierungen des Planungsproblems in Aussagenlogik. Eine empirische Evaluation des resultierenden Planers zeigte, dass er signifikant bessere Ergebnisse als der bisherige State-of-the-art im HTN-Planen zeigt. Weiterhin modifizierten wir den Planer so, dass er garantiert optimale Pläne finden kann. Anschließend wurden die von einem Nutzer geäußerten Wünsche zur Änderung eines Planes untersucht. Es wurde zunächst Lineare Temporal Logik als eine geeignete Repräsentation für diese Wünsche identifiziert. Anschließend wurde die Berechnungskomplexität der Beachtung dieser Wünsche während der Generierung von Handlungsplänen untersucht. Letztlich wurde gezeigt, wie diese als LTL-Formeln ausgedrückten Wünsche in den Planungsprozess eingebunden werden können. Dabei wurde die bisher für das Planen bekannte Kodierung von LTL in Aussagenlogik verbessert. Die entwickelten Techniken wurden in der praktischen Anwendung erprobt. Gemeinsam mit der Robert Bosch GmbH wurde der planungsbasierte Assistent ROBERT entwickelt, der Heimwerker bei Do-It-Yourself Projekten unterstützt. Mithilfe der entwickelten Planungstechnologien kann ROBERT seine Anweisungen in Realzeit auf die konkrete Situation des Nutzers anpassen und vorhandene Materialien und Werkzeuge optimal verwenden. Außerdem kann ROBERT die Wünsche und Anweisungen des Nutzers in die gegebenen Anweisungen mit einbeziehen.

Literaturverzeichnis

[Al16] Alford, Ron; Behnke, Gregor; Höller, Daniel; Bercher, Pascal; Biundo, Susanne; Aha, David W.: Bound to Plan: Exploiting Classical Heuristics via Automatic Translations of Tail-Recursive HTN Problems. In: ICAPS. 2016.

[BB18] Behnke, Gregor; Biundo, Susanne: X and more Parallelism: Integrating LTL-Next into SAT-based Planning with Trajectory Constraints While Allowing for Even More Parallelism. Inteligencia Artificial, 21(62):75–90, 2018.

[Be16] Behnke, Gregor; Höller, Daniel; Bercher, Pascal; Biundo, Susanne: Change the Plan – How hard can that be? In: ICAPS. 2016.

[Be17] Bercher, Pascal; Behnke, Gregor; Höller, Daniel; Biundo, Susanne: An Admissible HTN Planning Heuristic. In: IJCAI. 2017.

[Be18] Behnke, Gregor; Schiller, Marvin; Kraus, Matthias; Bercher, Pascal; Schmautz, Mario; Dorna, Michael; Minker, Wolfgang; Glimm, Birte; Biundo, Susanne: Instructing Novice Users on How to Use Tools in DIY Projects. In: IJCAI-ECAI. 2018.

[Be19a] Behnke, Gregor: Hierarchical planning through propositional logic: highly efficient, versatile, and flexible. dissertation, Ulm University, 2019.

[Be19b] Behnke, Gregor; Schiller, Marvin; Kraus, Matthias; Bercher, Pascal; Schmautz, Mario; Dorna, Michael; Dambier, Michael; Minker, Wolfgang; Glimm, Birte; Biundo, Susanne: Alice in DIY-Wonderland or: Instructing novice users on how to use tools in DIY projects. AI Communications, 32(1):31–57, 2019.

[Be20] Behnke, Gregor; Bercher, Pascal; Kraus, Matthias; Schiller, Marvin; Mickeleit, Kristof; Häge, Timo; Dorna, Michael; Dambier, Michael; Minker, Wolfgang; Glimm, Birte; Biundo, Susanne: New Developments for Robert – Assisting Novice Users Even Better in DIY Projects. In: ICAPS. 2020.

[BHB15] Behnke, Gregor; Höller, Daniel; Biundo, Susanne: On the Complexity of HTN Plan Verification and Its Implications for Plan Recognition. In: ICAPS. 2015.

[BHB17] Behnke, Gregor; Höller, Daniel; Biundo, Susanne: This is a solution! (... but is it though?) – Verifying solutions of hierarchical planning problems. In: ICAPS. 2017.

[BHB18a] Behnke, Gregor; Höller, Daniel; Biundo, Susanne: totSAT – Totally-Ordered Hierarchical Planning through SAT. In: AAAI. 2018.

[BHB18b] Behnke, Gregor; Höller, Daniel; Biundo, Susanne: Tracking Branches in Trees – A Propositional Encoding for solving Partially-Ordered HTN Planning Problems. In: ICTAI. 2018.

[BHB19a] Behnke, Gregor; Höller, Daniel; Biundo, Susanne: Bringing Order to Chaos – A Compact Representation of Partial Order in SAT-based HTN Planning. In: AAAI. 2019.

[BHB19b] Behnke, Gregor; Höller, Daniel; Biundo, Susanne: Finding Optimal Solutions in HTN Planning – A SAT-based Approach. In: IJCAI. 2019.

[BMC18] Barták, Roman; Maillard, Adrien; Cardoso, Rafael C.: Validation of Hierarchical Plans via Parsing of Attribute Grammars. In: ICAPS. 2018.

[By77] Byrne, Richard: Planning meals: Problem solving on a real data-base. Cognition, 1977.

[DGV13] De Giacomo, Giuseppe; Vardi, Moshe Y.: Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In: IJCAI. 2013.

[Dv14] Dvorak, Filip; Bit-Monnot, Arthur; Ingrand, Félix; Ghallab, Malik: A flexible ANML actor and planner in robotics. In: PlanRob. 2014.

[EHN94] Erol, Kutluhan; Hendler, James; Nau, Dana: UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning. In: AIPS. 1994.

[EHN96] Erol, Kutluhan; Hendler, James; Nau, Dana: Complexity results for HTN planning. Annals of Mathematics and AI, 18(1):69–93, 1996.

[GB11] Geier, Thomas; Bercher, Pascal: On the Decidability of HTN Planning with Task Insertion. In: IJCAI. 2011.

[Hö18] Höller, Daniel; Bercher, Pascal; Behnke, Gregor; Biundo, Biundo: A Generic Method to Guide HTN Progression Search with Classical Heuristics. In: ICAPS. 2018.

[Hö19] Höller, Daniel; Bercher, Pascal; Behnke, Gregor; Biundo, Susanne: HTN Planning as Heuristic Progression Search. JAIR, 2019.

[KS96] Kautz, Henry; Selman, Bart: Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In: AAAI. 1996.

[La83] Lamport, Leslie: What Good is Temporal Logic? In: Information Processing. 1983.

[MR07] Mattmüller, Robert; Rintanen, Jussi: Planning for Temporally Extended Goals as Propositional Satisfiability. In: IJCAI. 2007.

[Na03] Nau, Dana; Au, Tsz-Chiu; Ilghami, Okhtay; Kuter, Ugur; Murdock, J.; Wu, Dan; Yaman, Fusun: SHOP2: An HTN Planning System. Journal of Artificial Intelligence Research (JAIR), 20:379–404, 2003.

[Pn77] Pnueli, Amir: The Temporal Logic of Programs. In: SFCS. 1977.

[RHN06] Rintanen, Jussi; Heljanko, Kejio; Niemelä, Ilkka: Planning as satisfiability: parallel plans and algorithms for plan search. Artificial Intelligence, 170(12-13):1031–1080, 2006.

[Sm12] Smith, David E.: Planning As an Iterative Process. In: AAAI. 2012.

[WM05] Ward, Geoff; Morris, Robin: Introduction to the psychology of planning. Kapitel 1, 2005.

Gregor Behnke ist Jahrgang 1992. Nach dem Abitur im Jahr 2008 hat er bis 2014 an der Universität Rostock Informatik studiert. Er erwarb dort sowohl einen Bachelor als auch einen Master of Science. Nach seinem Studium war er wissenschaftlicher Mitarbeiter am Institut für Künstliche Intelligent der Universität Ulm von Prof. Dr. Susanne Biundo-Stephan. Er arbeitete dort im Rahmen des SFB/TRR 62 „Eine Companion-Technologie für kognitive technische Systeme“ am Hierarchischen Planen und dessen Einsatz in gemischt-initiativen Planungssystemen. Weiterhin war er im Transferprojekt „Do it yourself, but not alone: Companion-Technologie für die Heimwerkerunterstützung“ beschäftigt und wendete dort die von ihm entwickelten Planungstechnologien zur Unterstützung von Heimwerkern an. Im Dezember 2019 schloss er seine Promotion an der Universität Ulm mit der Note *summa cum laude* ab. Seit Januar 2020 ist er als PostDoc wissenschaftlicher Mitarbeiter am Lehrstuhl für Grundlagen der Künstlichen Intelligenz von Prof. Dr. Bernhard Nebel an der Albert-Ludwigs-Universität Freiburg.