Załącznik 3.1

Autoreferat

15.04.2019

1. Imię i Nazwisko

Radosław Klimek

2. Posiadane dyplomy i stopnie naukowe

Posiadane dyplomy, stopnie naukowe – z podaniem nazwy, miejsca i roku ich uzyskania oraz tytułu rozprawy doktorskiej.

— **Doktor nauk technicznych** w dyscyplinie Informatyka, Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki, rozprawa pt. *Warstwowa weryfikacja systemów współbieznych z wykorzystaniem logiki temporalnej* (praca wyróżniona), 1998.

— **Magister inżynier** informatyki, Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, Wydział Elektrotechniki, Automatyki i Elektroniki, 1987.

3. Informacje o dotychczasowym zatrudnieniu

Informacje o dotychczasowym zatrudnieniu w jednostkach naukowych:

— adiunkt, AGH, od 1998– aktualnie

— asystent, AGH, 1993–1998

— st. wykładowca, Instytut Politechniczny, PWSZ, Tarnów, 2006– aktualnie

— wykładowca, Instytut Politechniczny, PWSZ, Tarnów, 2002–2006

— adiunkt, WSTE, Sucha Beskidzka, 2007–2012

— dziekan, WSTE, Sucha Beskidzka, 2005–2009

— profesor WSTE, WSTE, Sucha Beskidzka, 2002–2007

4. Wykazanie osiągnięcia

4.1. Tytuł osiągnięcia naukowego

Cykl 10 publikacji: **Formalna weryfikacja modeli oprogramowania i projektowanie procesów decyzyjnych z zastosowaniem automatycznego wnioskowania**

4.2. Wykaz publikacji stanowiących osiągniecie naukowe

Wykaz publikacji (autor/autorzy, tytuł/tytuły publikacji, rok wydania, nazwa wydawnictwa):

[A1] R. Klimek, **Towards formal and deduction-based analysis of business models for SOA processes**. In *Proceedings of 4th International Conference on Agents and Artificial Intelligence (ICAART 2012), 6–8 February, 2012, Vilamoura, Algarve, Portugal*, J. Filipe and A. Fred (Eds.), vol. 2, pp. 325–330. SciTePress 2012. DOI=10.5220/0003740503250330. (**Web of Science**.)

[A2] R. Klimek, **Deduction-based formal verification of requirements models with automatic generation of logical specifications**. In *7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012, Revised Selected Papers), 29–30 June, 2012, Wrocław, Poland*, L. Maciaszek and J. Filipe (Eds.), (Communications in Computer and Information Science, vol. 410), pp. 157–171. Springer-Verlag 2013. DOI=10.1007/978-3-642-45422-6_11. (**Web of Science**.)

[A3] R. Klimek, **From extraction of logical specifications to deduction-based formal verification of requirements models**. In *Proceedings of 11th International Conference on Software Engineering and Formal Methods (SEFM 2013), 25–27 September 2013, Madrid, Spain*, R.M. Hierons, M.G. Merayo, and M. Bravetti (Eds.), (Lecture Notes in Computer Science, vol. 8137), pp. 61–75. Springer Verlag 2013. DOI=10.1007/978-3-642-40561-7_5. (**Web of Science**.)

[A4] R. Klimek, **A system for deduction-based formal verification of workflow-oriented software models**. *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 4, pp. 941–956, 2014. DOI=10.2478/amcs-2014-0069. (**Web of Science. IF (2014)=1,227; Quartiles (2014): Q1=Applied mathematics, Q3=Computer science, artificial intelligence**).

[A5] R. Klimek, **Pattern-based and composition-driven automatic generation of logical specifications for workflow-oriented software models**. *Journal of Logical and Algebraic Methods in Programming (Elsevier)*, vol. 104, pp. 201–226, 2019. DOI=10.1016/j.jlamp.2019.02.005. (**Web of Science, IF (2019)=0,634, Quartiles (2019): Q2=Logic, Q4=Computer science, Theory & methods**.)

[A6] R. Klimek, G. Rogus, **Modeling context-aware and agent-ready systems for the outdoor smart lighting**. In *Proceedings of 13th International Conference on*

Arificial Inteligence and Soft Computing (ICAISC 2014), 1–5 June, 2014, Zakopane, Poland, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L.A. Zadeh, and J.M. Zurada (Eds.), (Lecture Notes in Artificial Intelligence, vol. 8468), pp. 269–280. Springer Verlag 2014. DOI=10.1007/978-3-319-07176-3_23. (**Web of Science**.)

[A7] R. Klimek, L. Kotulski, **Proposal of a multiagent-based smart environment for the IoT**. In *Proceedings of the 10th International Conference on Intelligent Environments, Shanghai, China, 30th June–1st of July 2014, Workshop*, J.C. Augusto and T. Zhang (Eds.), (Ambient Intelligence and Smart Environments, vol. 18), pp. 37–44. IOS Press 2014. DOI=10.3233/978-1-61499-411-4-37. (**Web of Science**.)

[A8] R. Klimek, **Behaviour recognition and analysis in smart environments for contextaware applications**. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2015), October 9–12, 2015, City University of Hong Kong, Hong Kong*, pp. 1949–1955. IEEE Computer Society 2015. DOI=10.1109/SMC.2015.340. (**Web of Science**.)

[A9] R. Klimek, **Exploration of human activities using message streaming brokers and automated logical reasoning for ambient-assisted services**. *IEEE Access*, vol. 6, pp. 27127–27155, 2018. DOI=10.1109/ACCESS.2018.2834532. (**Web of Science, IF (2018)=3,557; Quartiles (2018): Q1=Computer science, Information systems, Q1=Engineering, Electrical & Electronic, Q1=Telecommunications**.)

[A10] R. Klimek, **Towards recognising individual behaviours from pervasive mobile datasets in urban spaces**. *Sustainability*, vol. 11, num. 6, 2019. DOI= 10.3390/su11061563. (**Web of Science, IF (2019)=2,075, Quartiles (2019): Q2=Environmental science, Q2=Environmental studies, Q2=Green & sustainable science & technology**.)

4.3. Omówienie celu naukowego

Omówienie ww. prac i osiągniętych wyników wraz z omówieniem ich ewentualnego wykorzystania[1].

4.3.1. Cel naukowy prac i osiągnięte wyniki – streszczenie rozszerzone

Ciągły i intensywny rozwój innowacyjnych technologii, zaawansowane wykorzystywanie procesów automatyzacji, przetwarznie olbrzymich zbiorów danych, bezprzewodowe transmisje danych, sieci sensorów, Internet rzeczy IoT (ang. *Internet of Things*), urzeczywistnienie wizji inteligentnej fabryki, a generalnie zanikanie bariery ludzie-maszyny,

[1]Na potrzeby cytowania prac zastosowano następujące oznaczenie jednoliterowym prefiksem: **A** – prace wchodzące w skład osiągnięcia naukowego będącego przedmiotem wniosku (tj. prace stanowiące przedstawiony cykl publikacji), **B** – prace inne, których autorem lub współautorem jest wnioskodawca, **C** – prace innych autorów.

wszystko w ramach tzw. *Czwartej rewolucji przemysłowej* (Industry 4.0), stawia to olbrzymie wyzwania w odniesieniu do procesów wytwórczych inżynierii oprogramowania, czy projektowania procesów monitorowania i podejmowania decyzji. Automatyzacja procesów formalnej weryfikacji oraz procesów decyzyjnych wpływa pozytywnie na jakość produktów i usług, wydajność systemów oraz wyższy poziom ich bezpieczeństwa, poprzez eliminowanie błędów bardzo trudnych lub wręcz niemożliwych do wykrycia, np. drogą testowania. Dotychczasowe metody są często mało efektywne. Z drugiej strony, dynamiczny rozwój systemów automatycznego wnioskowania, podejście logiczne SAT, stwarza nowe możliwości i wyzwania wpływając na obszary badawcze. Wśród tych wyzwań jest problem automatycznego i wiarygodnego pozyskiwania specyfikacji logicznej, która będzie stanowić dane wejściowe dla istniejących i dynamicznie rozwijających się systemów wnioskujących. Narzędzia wnioskowania dedukcyjnego mogą być wykorzystane zarówno w formalnej weryfikacji oprogramowania, jak i w podejmowaniu decyzji złożonych systemów.

Kluczowym osiągnięciem habilitanta jest stworzenie formalnych podstaw zaproponowanej metody ΠBCD (ang. *Pattern-based and Composition-driven*, w skrócie także ΠC), pozwalającej na projektowanie, w oparciu o przyjęte wzorce behawioralne oprogramowania, dowolnie złożonego modelu zachowania systemu informatycznego, a następnie wygenerowanie dla niego ekwiwalentnej specyfikacji logicznej. Zostały zdefiniowane formalnie, na bazie struktury Kripke'go, przykłady wzorców behawioralnych opartych na sieciach działań. Następnie dla nich zostały zdefiniowane, a także dla innych przypadków, wzorce logiczne zachowania w terminach rachunku zdań liniowej logiki temporalnej PLTL (ang. *Propositional Linear Temporal Logic*). Zaproponowana metoda jest skalowalna w górę, w znaczeniu nie ograniczania liczby zadań wchodzących w skład pojedynczego wzorca behawioralnego oraz wzorca logicznego, ale także możliwości definiowania własnych i zupełnie nowych wzorców. Opracowany został algorytm generowania specyfikacji logicznej, w postaci zbioru formuł logiki temporalnej PLTL, dla dowolnego modelu zachowania oprogramowania. Generowanie odbywa się automatycznie. Zostało udowodnione, że algorytm jest poprawny, ponadto zachowuje zarówno spełnialność dla generowanej specyfikacji logicznej, jak i jej względną zupełność. Całość zaproponowanego podejścia cechuje się silną kompozycyjnością, która została celowo zaimplementowana, jako w praktyce jedyna metoda pozwalająca efektywnie zarządzać złożonością dużych systemów.

Wykorzystaniem ΠBCD jest zaproponowana metoda, zorientowana na formalną weryfikację, dla procesów inżynierii wymagań (ang. *Requirements Engineering*, w skrócie RE), w oparciu o język UML (ang. *Unified Modeling Language*). Obejmuje ona transformacje od diagramów przypadków użycia, poprzez scenariusze przypadków użycia, do diagramów aktywności, z których następnie generowana jest automatycznie specyfikacja logiczna z wykorzystaniem algorytmów ΠBCD. Procesy RE mają kluczowe znaczenie dla jakości i powodzenia całości procesów projektowych oprogramowania. Dla systemów o podwyższonych wymaganiach bezpieczeństwa (ang. *safety critical systems*) są one krytyczne. Przedstawiona metoda wpisuje się w koncepcję zaproponowanego przez habilitanta formalnego i zintegrowanego środowiska projektowego (ang. *Formal Integrated Development Environment*) klasy F-IDE, które może być celem dalszych badań. Został zaproponowany szereg przypadków najważniejszych procesów wnioskowania logicznego, w oparciu o podejście dedukcyjne, dla takiego środowiska.

Kolejnym etapem badawczym są interakcje silników wnioskujących SAT (ang. *SAT*

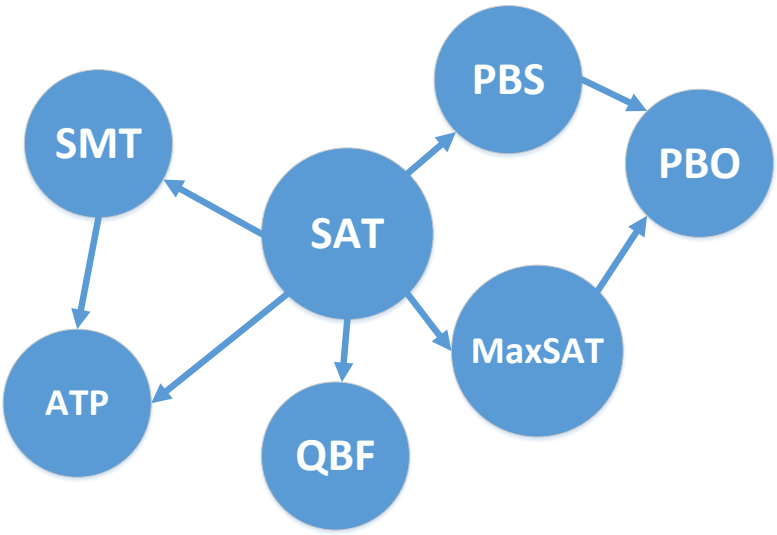
solvers) z systemami należącymi do inteligentnego środowiska (ang. *Intelligent Environments*, w skrócie IE), ale także działającymi w paradygmacie inteligencji otoczenia (ang. *Ambient Intelligence*, w skrócie AmI), oraz Internetu rzeczy IoT. Jako najbardziej zaawansowany należy wskazać opracowany system wspomagania pracy ratowników górskich, działający w koncepcji obliczeń rozpowszechnionych i wszechobecnych (ang. *pervasive and ubiquitous computing*). Podejmowanie decyzji w tym systemie odbywa się na bazie wnioskowania logicznego. Wykrywane są sytuacje zagrożenia turystów przebywających w monitorowanym obszarze, po przeanalizowaniu dużej ilości danych pochodzących zarówno z rozlokowanych w terenie stacji pogodowych jak i ogólnodostępnych stacji telefonii komórkowej. Została wykazana wykonalność i efektywność takiego systemu, przetwarzającego duże strumienie danych z wykorzystaniem brokerów informacji oraz silników SAT. Pomyślnie przetestowano kilka brokerów i ich konfiguracje oraz wiele silników wnioskujących.

Została także zaproponowana metoda konstruowania trajektorii przemieszczania się obiektów, np. turystów, w monitorowanym obszarze miejskim, na podstawie danych pochodzących ze stacji telefonii komórkowej (ang. *Base Transiver Station*, w skrócie BTS). Jest to metoda uniwersalna, w której może być wykorzystane wnioskowanie logiczne. Został zaproponowany system wielo-agentowy, generujący trajektorie indywidualne, które są wzbogacane o informacje o wybranych elementach infrastruktury miejskiej, na które natrafił monitorowany obiekt. Tak uzyskane trajektorie mogą być następnie poddane odrębnej analizie celem stworzenia charakterystyki przemieszczania się badanego obiektu. Ciekawym i praktycznym obszarem zastosowania tej koncepcji jest profilowanie zachowań zwiedzających poruszających się w określonym rejonie turystycznym.

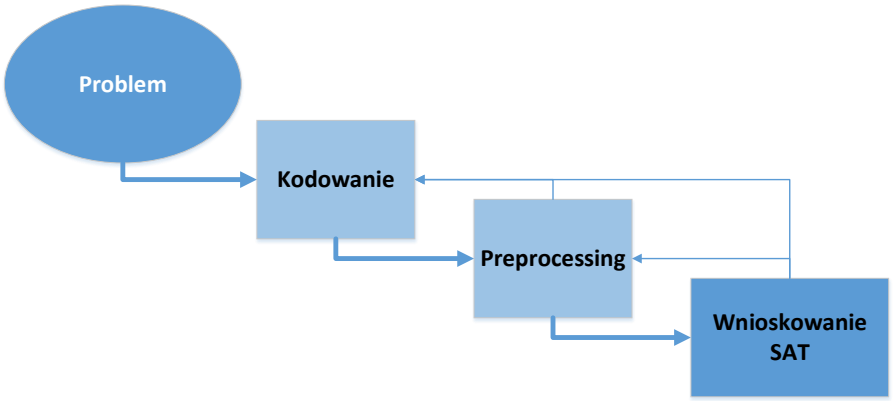
4.3.2. Osiągnięte wyniki – omówienie

Kontekst i cel badań W ostatnich latach nastąpił ogromny postęp w odniesieniu do systemów automatycznego wnioskowania, tzw. silników wnioskujących, a szczególnie spektakularny postęp dotyczy silników SAT (ang. *SAT solvers*). Donald Knuth w swojej znanej i fundamentalnej monografii *The Art of Computer Programming* dołączył entuzjastyczny rozdział [C24] dotyczący problemu spełnialności, przełomowej metody znajdowania podstawień spełniających opartej na metodzie uczenia się kierowanego konfliktami CDCL (ang. *Conflict-Driven Clause Learning*), na której dziś bazują najefektywniejsze silniki SAT. Dobrym udokumentowaniem tego postępu jest także rozdział [C25], dotyczący podstawowych technik i proponowanych rozwiązań dla silników SAT, podstawowa monografia dla problemu spełnialności [C26], jak również liczne strony webowe, gromadzące zarówno społeczność badaczy, jak i wszystkich zainteresowanych, którzy poszukują dobrych źródeł informacji na wymienione tematy, np. por. [C27] dla systemów automatycznego dowodzenia twierdzeń, [C28] dla klasycznych silników SAT, ale i systemów pokrewnych, czy [C29] dla przeprowadzanych corocznie zawodów dla silników SAT i systemów pokrewnych. Można więc powiedzieć, że sytuacja dziś jest o tyle obiecująca, że problem który jest uznany za klasyczny przykład zagadnienia NP-zupełnego, poprzez zaproponowane liczne heurystyki, w tym podstawową CDCL, pozwolił na zapanowanie, w wielu przypadkach, nad złożonością obliczeniową problemów. Obecnie zadania z ok. 50 tysiącami zmiennych są rozwiązywane rutynowo na przeciętnej i ogólnodostępnej maszynie obliczeniowej [C29]. Problemy większe, powiedzmy rzędu ok. sto tysięcy lub milion zmiennych, są również

osiągalne, ale ich czas rozwiązania zależy od struktury wewnętrznej problemu.



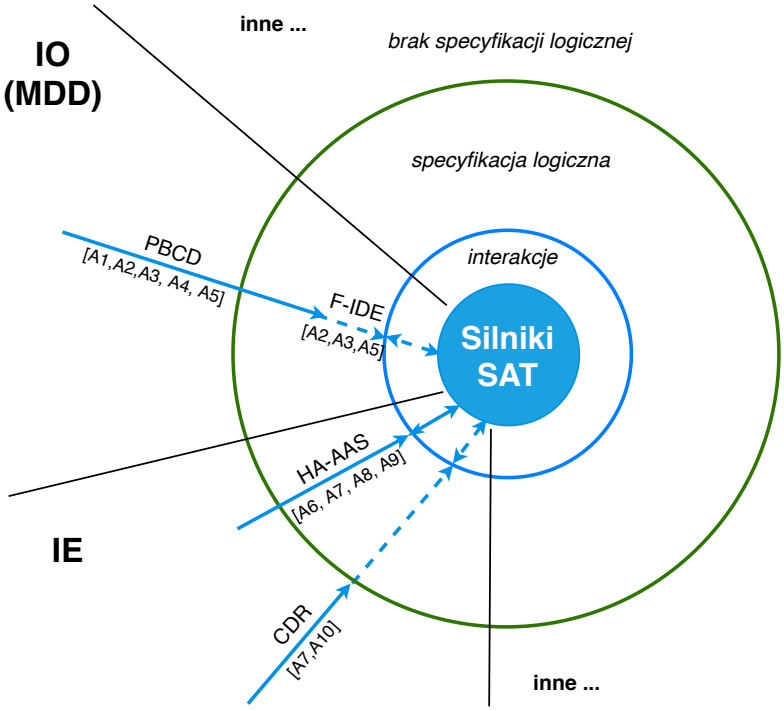
Rysunek 1: Rozwój systemów wnioskowania automatycznego i ich wzajemne oddziaływanie. SAT – klasyczne silniki SAT na bazie metody CDCL (ang. *Conflict-Driven Clause Learning*), SMT – klasyczne silniki SAT powiązane z wybraną teorią pierwszego rzędu (ang. *Satisfiability Modulo Theories*), ATP – automatyczne dowodzenie twierdzeń (ang. *Automated Theorem Proving*), QBF – kwantyfikowane formuły logiczne (ang. *Quantified Boolean Formulas*), MaxSAT – problem maksymalnej spełnialności (ang. *Maximum Satisfiability*), PBS – spełnialność pseudo-Boolowska (ang. *Pseudo-Boolean Satisfiability*), PBO – optymalizacja pseudo-Boolowska (ang. *Pseudo-Boolean Optimizers*)



Rysunek 2: Łańcuch działań w przejściu od sformułowania problemu do jego rozwiązania, por. [C30]

Wspomniany rozwój systemów rozwiązujących klasyczny problem spełnialności rzutuje, ale i w różnym stopniu wpływa, także na inne obszary związane z automatycznym wnioskowaniem, por. rysunek 1. Dostępnych jest dzisiaj wiele różnych systemów wnioskujących. Interakcje z silnikami SAT, oraz innymi systemami, są ważnym obszarem wielu

badań. Jednak na całość zagadnienia należy też patrzeć w szerszym kontekście, por. rysunek 2, gdyż duży postęp dokonuje się obecnie także w odniesieniu do preprocessingu. Dokładne omówienie tej problematyki przekracza cel i rozmiary tego opracowania, a pokazanie kontekstu badawczego prac wchodzących w skład przedstawionego cyklu można uznać za zakończone.



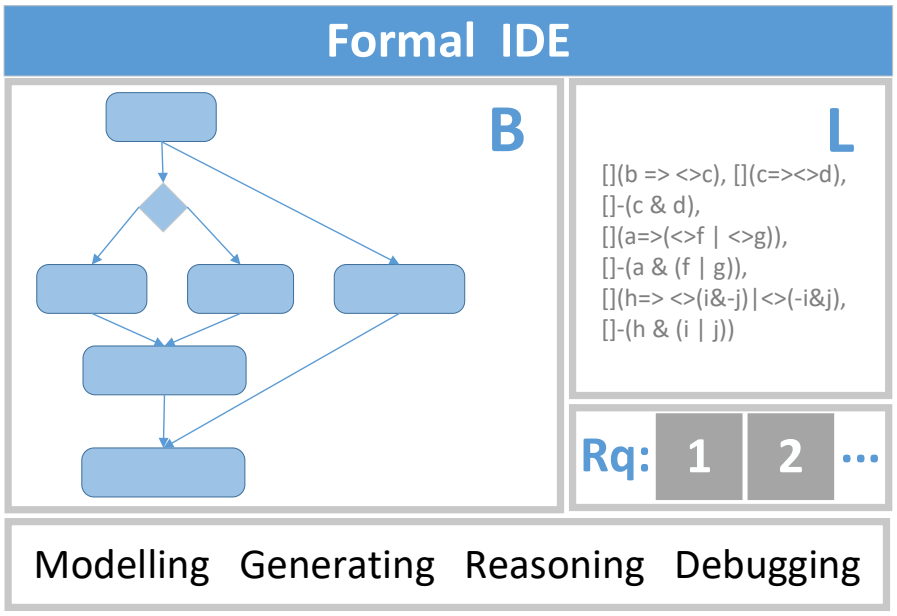
Rysunek 3: Prace badawcze cyklu zorientowane na interakcje z silnikami wnioskującymi, z podziałem na obszary badawcze: IO – inżynieria oprogramowania dla projektowania kierowanego modelami (ang. *Model-Driven Development*), IE – inteligentne środowiska, a także inne niewymienione obszary. Środek – silniki logiczne SAT – por. rysunek 1, niebieski okrąg – osiągnięcie pełnego zakodowania logicznego problemu, zielony okrąg – częściowe zakodowanie logiczne problemu, na zewnątrz brak specyfikacji logicznej. Strzałki z dwoma grotami – interakcje logiczne z silnikami wnioskującym, strzałki z jednym grotem – prace badawcze w kierunku budowania specyfikacji logicznej. Linie ciągłe – prace wykonane i wdrożone, linie przerywane – prace częściowo zrealizowane lub planowane.

Na rysunku 3 został pokazany zakres i kierunek prac badawczych cyklu, zmierzający, mówiąc nieformalnie, do logicznego zakodowania wybranych problemów i, docelowo, interakcji z silnikami wnioskującymi. Im bliżej środka – i przekraczania okręgów granicznych – tym wyższy stopień zakodowania problemu do specyfikacji logicznej. Główne wątki cyklu obejmują następujące tematy: ΠBCD – automatyczne generowanie specyfikacji logicznej dla modeli behawioralnych oprogramowania. F-IDE – zintegrowane środowisko projektowe (ang. *Formal Integrated Development Environment*) planowane do utworzenia, obecnie zaproponowana metoda rozwijania modeli inżynierii wymagań RE (ang. *Requirements Engineering*) dla takiego środowiska, wraz z bogatymi scenariuszami

analiz poprawnościowych i wykrywania wzajemnych relacji logicznych. HA-AAS – pełne zakodowanie problemu podejmowania decyzji w serwisach wspomaganych otoczeniem, w odniesieniu do zagadnienia rozumienia ludzkich aktywności (ang. *Human Activities*, *Ambient-Assisted Services*), i predykcji zagrożeń, wraz ze zrealizowaną interakcją z silnikami SAT. CDR – rekonstruowanie indywidualnych trajektorii dla przemieszczających się w monitorowanym obszarze mieszkańców, na podstawie danych z rekordów CDR (ang. *Call Detail Record*), w następnym kroku planowanie jest zbudowanie specyfikacji logicznej. Podsumowując z innej perspektywy: w przypadku IIBCD generowana jest złożona specyfikacja logiczna z początkowo abstrakcyjnych modeli behawioralnych, pozostaje jeszcze zbudowanie środowiska F-IDE, aby w ten sposób doprowadzić do interakcji z silnikami logicznymi. Dla HA-AAS zbudowano pełną specyfikację logiczną i zrealizowano interakcję z silnikami wnioskującymi. Z kolei dla CDR zbudowano indywidualne trajektorie, na podstawie których można już budować specyfikacje logiczne, np. celem poznawania preferencji mieszkańców, i tym samym doprowadzić do interakcji z systemami wnioskowania logicznego.

Inżynieria oprogramowania ([A1, A2, A3, A4, A5]) Formalna analiza, modelowanie i weryfikacja modeli behawioralnych/zachowania (ang. *model behaviours*) systemów oprogramowania, z wykorzystaniem aparatu logiki formalnej, z uwzględnieniem procesów wnioskowania logicznego, jest jednym z dominujących, obok algebraicznego i grafowego, podejść w formalnej inżynierii oprogramowania [C31]. Odnotowuje się postęp, por. np. [C32], w odniesieniu do automatycznego wnioskowania w celu weryfikacji różnych modeli informatycznych. Jednak integracja procesów wytwórczych oprogramowania, a przede wszystkim analizy i modelowania zachowania systemów, z silnikami wnioskującymi jest dużym wyzwaniem, a powodzenie w tym zakresie umożliwiałoby formalną weryfikację coraz bardzej złożonych modeli. Szczególnie obiecujący kierunek, to zintegrowane środowiska Formal IDE (ang. *Formal Integrated Development Environment*) lub F-IDE, por. np. [C33]. Środowiska takie mogą oferować przyjazne dla analityka i projektanta oprogramowania narzędzia, łączące w sobie wizualizację modeli, generatory specyfikacji logicznych, logiczne dowodzenie własności z wykorzystaniem wbudowanych silników SAT lub systemów dowodzenia twierdzeń. Wszystko obsługiwane poprzez wspólny interfejs użytkownika i wygodne menu. Strategicznym celem przedstawionego cyklu prac jest stworzenie takiego właśnie narzędzia klasy F-IDE, por. [A5, rysunek 1].

Na rysunku 4 pokazano przykładowe środowisko klasy F-IDE, które w przyszłości będzie zbudowane. W pewnym zakresie było ono już dyskutowane w pracach [A2, A3, A4], a silne podstawy teoretyczne dostarcza do niego praca [A5]. Pozwala to na badanie licznych relacji logicznych zaprojektowanego przez inżyniera modelu zachowania oprogramowania. Dla zaprojektowanego modelu jest generowana w locie, lub też alternatywnie na żądanie, równoważna specyfikacja logiczna. Może być ona wstępnie poddana badaniu na spełnialność. Po uwzględnieniu dodatkowych i wprowadzonych przez analityka formuł, także szereg innych własności logicznych, uwzględniających np. spójność, niesprzeczność, żywotność, bezpieczeństwo, rozłączność, ale i inne, por. [A5, podrozdział 6.4], gdzie podano liczne scenariusze dla takich procesów wnioskowania logicznego. Dokładne przedstawienie działania środowiska, jego szczegółowej architektury, stosowanych procedur, będzie przedmiotem kolejnych badań. Oczywiste jest jednak, że zasadniczą rolę w tym środowisku odgrywa generator specyfikacji logicznej działający w oparciu o stworzone w cyklu



Rysunek 4: Przykładowe środowisko F-IDE do analizy, rozwijania i weryfikowania modeli oprogramowania, gdzie B – model behawioralny, L – specyfikacja logiczna modelu, przy czym $\Pi C : B \to L$, por. algorytm 1, Rq – kolejne formuły poprawnościowe dla wygenerowanego modelu logicznego, takie, że $L \models Rq$, por. także [A5, podrozdział 6.4, tabela 3, tabela 4]. Całe środowisko pracuje w cyklu: Modelling (B) $\to$ Generating (ΠC) $\to$ Reasoning (L) $\to$ Debugging.

prac podstawy teoretyczne, a konkretnie metodę ΠBCD.

Konieczność posiadania specyfikacji logicznej dla modelu zachowania projektowanego systemu, a mówiąc ściślej zakodowania modelu zachowania oprogramowania w formuły logiczne, jest więc oczywista. Specyfikacja taka stanowi podstawę dalszych, bogatych analiz logicznych. Jednak budowanie takiej specyfikacji w sposób manualny, przekracza możliwości przeciętnego i niedoświadczonego inżyniera IT. Ponadto, nigdy nie byłby on pewny poprawności tak wytworzonej specyfikacji. Manualne tworzenie specyfikacji jest procesem żmudnym i podatnym na błędy, a wielokrotne powtarzanie tej czynności, przy jakiejkolwiek modyfikacji modelu, jest nieefektywne. Stąd też potrzeba automatyzacji tego procesu jest zrozumiała, uzasadniona i szczególnie ważna, a dla planowanego F-IDE wręcz kluczowa. Obszerne badania literaturowe pokazują [A4, A5], że nie są prowadzone badania w tym zakresie, szczególnie w odniesieniu do logiki temporalnej [B11], także w odniesieniu do modeli zachowania opartych na przepływach/sieciach działań, jak i wzorcach projektowych. Przedstawiony cykl prac proponuje całościowe rozwiązanie tego problemu. Patrząc chronologicznie, zostały rozpoczęte prace w odniesieniu do wzorców SOA [A1] (ang. *Service Oriented Architecture*), następnie były dyskutowane diagramy aktywności języka UML [C34] (ang. *Unified Modeling Language*) dla inżynierii wymagań RE [A3], wreszcie sieci działań BPMN [A4] (ang. *Business Process Modeling Notation*). Kulminacyjnym momentem badań jest praca [A5], gdzie zaproponowano metodę opartą na wzorcach behawioralnych i logicznych, oraz kierowaną kompozycyjnością ΠBCD

(ang. *Pattern-based Composition-driven*, lub krótko ΠC). Generuje ona automatycznie i na żądanie specyfikację logiczną, wyrażoną w rachunku zdań liniowej logiki temporalnej PLTL (ang. *Propositional Linear Temporal Logic*), po przeanalizowaniu modelu zachowania oprogramowania pod kątem użytych wzorców behawioralnych, które mogą tworzyć praktycznie dowolnie złożoną sieć działań.

Założenia zaproponowanej metody ΠBCD są następujące [A3, A4, A5]. Przyjmuje się, że cały projektowany model zachowania może być tworzony tylko z wykorzystaniem predefiniowanych wzorców behawioralnych, utworzonych na bazie sieci działań. O ile taki podstawowy zbiór wzorców behawioralnych jest stały dla danego procesu projektowego, to należy wyraźnie podkreślić, że sama metoda ΠBCD jest otwarta na dowolne i predefiniowane wzorce behawioralne. Nie ma szczególnego ograniczenia zarówno odnośnie wielkości pojedynczego wzorca, tj. liczby zadań tworzących wzorzec oraz powiązań pomiędzy poszczególnymi zadaniami, jak i ogólnej liczby przyjętych dla danego procesu projektowego wzorców. Sama metoda jest skalowalna w górę w odniesieniu do granularności wzorca, jak i liczby stosowanych w projektowaniu wzorców. Jedynym ograniczeniem od dołu jest żądanie aby wzorzec posiadał minimum dwa zadania. Ograniczenie od góry jest raczej zdroworozsądkowe: im większy będzie wzorzec behawioralny, tym więcej wysiłku będzie wymagać zdefiniowanie danego wzorca w terminach logiki PLTL. W proponowanej metodzie ΠBCD zostało przyjęte, że dla każdego wzorca behawioralnego jest pre-definiowany ekwiwalentny mu wzorzec logiczny. Ważną cechą metody jest możliwość wielokrotnie stosowania i wykorzystywania raz zdefiniowanego wzorca w dowolnym procesie projektowym. Zdefiniowanie wzorca w terminach logiki temporalnej dokonywane jest raz, np. przez doświadczonego logika gwarantującego poprawność logiczną wzorca. Logik w trakcie projekowania może się wspierać istniejącymi systemami wnioskowania logicznego, celem sprawdzenia logicznej spójności proponowanego wzorca logicznego. Następnie wzorzec ten może być wykorzystywany przez zwykłego analityka IT w procesie projektowym wielokrotnie, bez żadnych ograniczeń ilościowych ani czasowych.

Wspomniana możliwość posiłkowania się w trakcie procesu projektowego wzorca logicznego istniejącymi systemami wnioskującymi jest kolejną praktyczną zaletą metody – dobrze określone wzorce logiczne, względnie małe, łatwo poddają się analizie systemów wnioskowania logicznego, a wyniki tej analizy są zazwyczaj proste w interpretacji.

Podstawowym problemem, z punktu widzenia algorytmizacji i poprawności ΠBCD, jest wygenerowanie specyfikacji logicznej dla całej i złożonej sieci działań projektowanego systemu. Użyte wzorce behawioralne mogą być zagnieżdżane, co jest całkowicie naturalnym procesem projektowym, mogą być stosowane sekwencyjnie, mogą być komponowane współbieżnie, wreszcie wykonanie poszczególnych zadań może być uzależnione od spełnienia pewnych warunków logicznych. Tymczasem wynikowa specyfikacja logiczna dla całego modelu zachowania nie może być zwykłym, w znaczeniu teorio-mnogościowym, sumowaniem poszczególnych formuł logicznych, które składają się na użyte w procesie projektowym wzorce behawioralne, i w konsekwencji wzorce logiczne. Problemy te zostały rozwiązane w opracowanej metodzie.

Wyniki przedstawionego cyklu prac, oprócz zaproponowania koncepcji ΠBCD oraz wprowadzenia wzorców behawioralnych i logicznych do procesu projektowego, obejmują też precyzyjne zdefiniowanie semantyki wzorców w terminach struktury Kripke'go. Struktura Kripke jest powszechnie stosowanym formalizmem definiowania semantyki dla logik modalnych, jednak jak zauważono w pracy [A5], może być także stosowana w odniesie-

niu do sieci działań rozumianych jako wzorce behawioralne. Aby w pełni ją wykorzystać, w pracy [A5] powiązano koncepcję struktury Kripke z koncepcją przedziałów (ang. *interrvals*), por. np. [C35], co nie jest opisane w dostępnej literaturze, a co stanowi rozszerzenie znanej koncepcji struktury Kripke i pozwala lepiej i łatwiej oddać złożony charakter behawioralny wzorca składającego się z wielu wzajemnie oddziaływujących na siebie zadań. W pracy [A5] zdefiniowano w ten sposób przykładowe wzorce behawioralne dla wielu typowych schematów przetwarzania. Następnie zdefiniowano wzorce behawioralne w terminach logiki PLTL, uzyskując w ten sposób wzorce logiczne. Prace [A5, A4] podają łącznie kilkanaście definicji różnych wzorców, zarówno o przeznaczeniu ogólnym, jak i zorientowanych na język BPMN. W tabeli 1 pokazano tylko wybrane i przykładowe definicje wzorców i jest to raczej zbiór minimalny niezbędny przy założeniu operowania w hipotetycznym procesie projektowym następującymi zaaprobowanymi wzorcami, stanowiącymi (lokalnie) przyjęty kanon:

$$\Pi = \{Seq, SeqSeq, Cond, Para, Loop\} \tag{1}$$

(Nieformalnie znaczenie wzorców sugerują ich nazwy: sekwencja, podwójna sekwencja, instrukcja warunkowa, zrównoleglenie, pętla.) Rozszerzenie i modyfikacja listy wzorców jest zawsze możliwa, co pozwala na dopasowanie kanonu do indywidualnych potrzeb projektowch. W metodzie ΠBCD nie ma szczególnych ograniczeń odnośnie stosowania wzorców behawioralnych, a wskazane są jedynie pewne zalecenia wynikające z pragmatyki, por. [A5]. Dla przyjętego kanonu wzorców Π, por. formuła (1), definiuje się ustalony zbiór własności logicznych Σ, który będzie podstawą dalszych działań, w szczególności będzie stanowił dane wejściowe dla głównego algorytmu ΠBCD, por. rysunek 6.

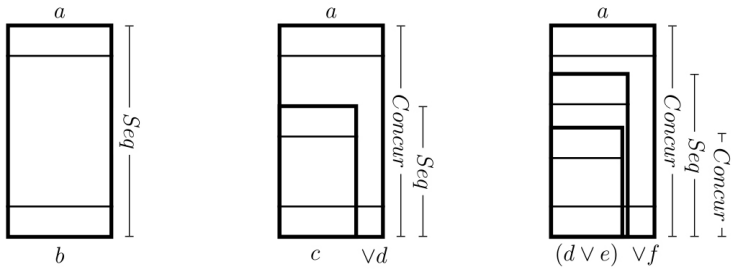
Tabela 1: Zbiór ustalonych własności logicznych Σ dla zaaprobowanych wzorców behawioralnych Π

$$\begin{aligned}\Sigma = \{\quad & \mathbf{Seq}(a1,a2) = \langle a1, a2, \Diamond a1, \Box(a1 \Rightarrow \Diamond a2), \Box\neg(a1 \wedge a2)\rangle,\\ & \mathbf{SeqSeq}(a1,a2,a3) = \langle a1, a3, \Diamond a1, \Box(a1 \Rightarrow \Diamond a2), \Box(a2 \Rightarrow \Diamond a3), \Box\neg(a1 \wedge a2), \Box\neg(a2 \wedge a3)\rangle,\\ & \mathbf{Cond}(a1,a2,a3,a4) = \langle a1, a4, \Diamond a1, \Box(a1 \Rightarrow (\Diamond a2 \wedge \neg\Diamond a3) \vee (\neg\Diamond a2 \wedge \Diamond a3)), \Box(a1^{+} \Rightarrow \Diamond a2), \Box(a1^{-} \Rightarrow \Diamond a3), \Box(a2 \vee a3 \Rightarrow \Diamond a4), \Box\neg(a1 \wedge (a2 \vee a3)), \Box\neg((a2 \vee a3) \wedge a4)\rangle,\\ & \mathbf{Para}(a1,a2,a3,a4) = \langle a1, a4, \Diamond a1, \Box(a1 \Rightarrow \Diamond a2 \wedge \Diamond a3), \Box(a2 \Rightarrow \Diamond a4), \Box(a3 \Rightarrow \Diamond a4), \Box\neg(a1 \wedge (a2 \vee a3)), \Box\neg((a2 \vee a3) \wedge a4)\rangle,\\ & \mathbf{Loop}(a1,a2,a3,a4) = \langle a1, a4, \Diamond a1, \Box(a1 \Rightarrow \Diamond a2), \Box(a2 \Rightarrow (\Diamond a3 \wedge \Diamond a4) \vee (\neg\Diamond a3 \wedge \Diamond a4)), \Box(a2 \wedge a2^{+} \Rightarrow \Diamond a3), \Box(a2 \wedge a2^{-} \Rightarrow \neg\Diamond a3 \wedge \Diamond a4), \Box(a3 \Rightarrow \Diamond a2), \Box(a4 \Rightarrow \neg\Diamond a2 \wedge \neg\Diamond a3), \Box\neg(a1 \wedge (a2 \vee a3 \vee a4)), \Box\neg(a2 \wedge (a1 \vee a3 \vee a4)), \Box\neg(a3 \wedge (a1 \vee a2 \vee a4)), \Box\neg(a4 \wedge (a1 \vee a2 \vee a3))\rangle\}\end{aligned}$$

Model zachowania projektowanego systemu dowolnie złożonego, ale zgodnie z założeniami o stosowaniu tylko predefiniowanych wzorców, wymaga reprezentacji pośredniej, tekstowej, która będzie stanowiła dane wejściowe dla algorytmu generującego specyfikację logiczną, por. rysunek 6. Zostało wprowadzone pojęcie wyrażenia logicznego [A4, A5]

(ang. *pattern expression*), które jest zbliżone do znanego w literaturze wyrażenia regularnego i pozwala na odwzorowanie dowolnego zagnieżdzenia wzorców zachowania użytych w projekcie systemu. W obu cytowanych pracach sformułowano i udowodniono twierdzenia, że każde dwa użyte wzorce w wyrażeniu są albo całowicie rozłączne, albo zawarte jeden w drugim. Podobnie zbiory atomowych argumentów dowolnych dwóch wzorców są albo zbiorami rozłącznymi, albo zawarte jedno w drugim. Te ważne stwierdzenia będą pomocne w algorytmie generowania specyfikacji logicznej.

Podstawowa trudność w wygenerowaniu specyfikacji logicznej dla (dowolnie) złożonej sieci działań wynika z faktu, że poszczególne wzorce behawioralne mogą być dowolnie zagnieżdżane. Co więcej, zagnieżdżanie takie może mieć miejsce – mówiąc nieformalnie – na początku danego wzorca, w jego środku lub na jego końcu. Każda z tych sytuacji jest odmienna i zasadniczo powinna być inaczej traktowana. Dla każdego wzorca definiujemy jego warunki początkowe oraz warunki końcowe, które opisują, w terminach logiki formalnej, sytuację gdy sterowanie jest przekazywane, odpowiednio, do wzorca oraz opuszcza wzorzec, albo inaczej to samo: inicjuje oraz finalizuje wzorzec. W przypadku zagnieżdżeń, i sytuacji jakby nakładania się warunków początkowych lub końcowych, budujemy na drodze algorytmicznej skonsolidowane wyrażenie początkowe i skonsolidowane wyrażenie końcowe. W pracy [A5] zostały wprowadzone wymienione koncepcje, a także przedstawiono algorytm ich wyznaczania dla dowolnej sytuacji projektowej, tj. dowolnego zagnieżdżenia. Rysunek 5 pokazuje proste przykłady sytuacji konsolidacji. W rzeczywistości rozpatrujemy konsolidację zarówno w odniesieniu do warunków początkowych jak i końcowych. Należy pokreślić, że wyznaczanie skonsolidowanego wyrażenia można dowolnie skalować w górę, tj. w kierunku coraz bardziej złożonych sytuacji projektowych. Proces i koncepcja konsolidowania wyrażeń logicznych jest najlepiej pokazana w pracy [A5].

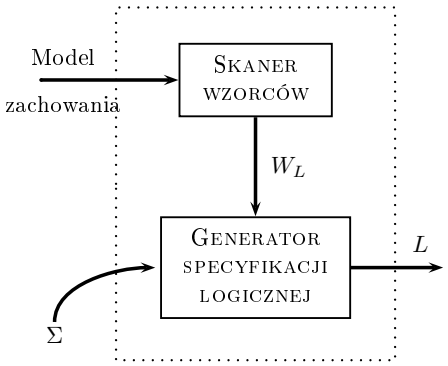


Rysunek 5: Proste przykłady wyrażeń skonsolidowanych [A5]

Specyfikacja logiczna dla dowolnego projektu sieci działań jest rozumiana, w znaczeniu pracy [A5], jako zbiór formuł logiki temporalnej wygenerowanych algorytmem ΠC

$$L(W_L) = \{f_i : i > 0 \wedge f_i \in \Pi C(w, \Sigma)\} \tag{2}$$

dla każdego wyrażenia $w \in W_L$, gdzie w jest dowolnym wyrażeniem logicznym należącym do pewnego języka wyrażeń W_L, a każda f_i jest poprawną składniowo formułą rachunku zdań logiki temporalnej PLTL. Natomiast algorytm generowania został pokazany jako Algorytm 1, a schemat i kontekst działania całego systemu na rysunku 6. Pominięto tutaj pewne algorytmy wykonujące obliczenia wstępne (preprocessing), które zamieszczono w pracy [A5].



Rysunek 6: Generowanie specyfikacji logicznej dla modeli oprogramowania [A5]

W pracy [A5] zostało pokazane, że algorytm 1 zawsze zatrzyma się, to znaczy nie ulegnie zapętleniu, a jego złożoność obliczeniowa, jak i towarzyszących mu dwóch innych algorytmów mających za zadanie realizować pewne obliczenia wstępne, wynosi $\Theta(p)$, gdzie p to liczba wzorców użytych w całym wyrażeniu. W cytowanej pracy zostały pokazane także inne własności algorytmu, a ciekawym stwierdzeniem jest to, że algorytm w trakcie generowania specyfikacji logicznej rozpina każde złożone wyrażenie wejściowe do postaci drzewa, gdzie poszczególne węzły zawierają użyte w wyrażeniu, i kolejno przetwarzane, wzorce. Tylko liście drzewa zawierają wzorce nie posiadające zagnieżdżeń. Z tego wynika, że dla zbioru wyrażeń wejściowych, składających się na cały model zachowania projektu, otrzymujemy las drzew.

Zaproponowana metoda ΠBCD cechuje się silną kompozycyjnością. Jest to celowe dążenie implementowane w pracach składających się na przedstawiony cykl, ale też innych pracach habilitanta, gdyż tylko w ten sposób możliwe jest sformułowanie zunifikowanego narzędzia, obejmującego zaaprobowane wzorce behawioralne i logiczne, a także algorytmy prowadzące do wygenerowania formalnej specyfikacji wyrażonej w logice temporalnej. Narzędzie takie będzie praktycznie użyteczne dla inżynierów IT w ich pracach analitycznych i projektowych nad modelami zachowania oprogramowania. Takie podejście pozwala na automatyzację procesu generowania specyfikacji logicznej. Stavros Tripakis [C36, C37] argumentuje za kompozycyjnością jako kluczowym podejściem do zarządzania złożonością dużych systemów, a także zarządzania dowolnymi zmianami w projekcie i systemie.

Jak stwierdzono w pracy [A5], zaproponowana w cyklu prac kompozycyjność cechuje się następującymi własnościami:

— *produktywnością* – rozumiana jako nieograniczona możliwość użycia systemu do wyrażania nowych rzeczy, także otwartość lub kreatywność, innymi słowy, umiejętność rozumienia nieznanych wcześniej zdań złożonych, także tworzenie nieskończenie wielu złożonych zdań;

— *systematycznością* – jakość wynikająca z uporządkowania i powtarzalności, także przeciwstawienie się chaotycznemu porządkowi, innymi słowy, fakt, że istnieją stałe i zrozumiałe wzorce, które są używane do budowania bardziej złożonych zdań;

— *uczeniem się* – jakość wynikająca ze studiowania, także użyteczność nauczenia się

Algorytm 1 Generowanie specyfikacji logicznej (ΠC), por. [A5]

Input: wyrażenie logiczne w
Input: niepusty zbiór predefiniowanych wzorców Σ
Output: logiczna specyfikacja L

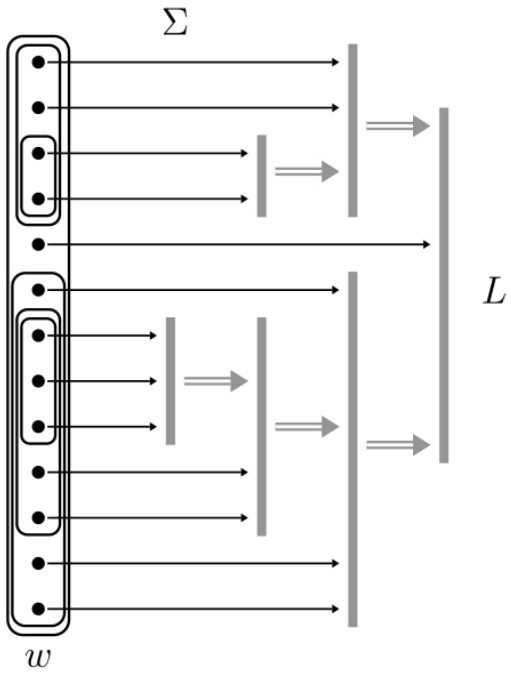
1: L:= $\emptyset$; ▷ inicjowanie specyfikacji
2: lab:= $Labelling(w)$;
3: **for** l:= $max(lab)$ **downto** 1 **do**
4: c:=1; ▷ bieżący wzorzec o etykiecie l do pobrania
5: p:= $getPat(lab, l, c)$; ▷ kolejny wzorzec najbardziej z lewej o etykiecie l
6: **repeat**
7: $L2$:= $\Sigma.p.pat()$; ▷ formuły PLTL z Σ dla p
8: **for** j:=1 **to** $|p|$ **do** ▷ każdy argument we wzorcu p
9: **if** $p \uparrow j$ is non-atomic **then** ▷ wtedy a_j jest sam wzorcem
10: $cons$:= $ConsEx(p \uparrow j, ini, \Sigma)$ + "$\vee$" +
11: $ConsEx(p \uparrow j, fin, \Sigma)$;
12: replace in $L2$ every $p \uparrow j$ by $cons$
13: **end if**
14: **end for**
15: $L := L \cup L2$;
16: c ++;
17: p:= $getPat(lab, l, c)$ ▷ następny wzorzec dla etykiety l
18: **until** $p = \varepsilon$ ▷ nie ma już kolejnych wzorców
19: **end for**
20: **return** L

korzystania z czegoś, innymi słowy, posiadanie procedur zdobywania wiedzy, które pozwalają osiągnąć właściwy język.

Takie scharakteryzowanie kompozycyjności w cyklu prac stanowi też odniesienie własne do kompozycyjności zaproponowanej w uznanych pracach [C38, C39]. Ponadto, w pracy [A5] została finalnie zdefiniowana kompozycyjność w terminach wprowadzanych we wcześniejszych pracach składających się na przedstawiony cykl, oraz innych pracach habilitanta, co z kolei nieformalnie pokazano na rysunku 7.

Fundamentalną i wymagającą odpowiedzi kwestią w przypadku generowania specyfikacji logicznej algorytmem ΠC są pytania dotyczące własności logicznych tak uzyskanej specyfikacji [A5, A4]. Określono to następująco. Jeżeli τ jest zbiorem syntaktycznych transformacji, obejmujących m.in. aksjomaty, twierdzenia, podstawienia i inne operacje, pozwalające na przekształcenie jednych formuł w inne formuły, a także jeśli κ jest semantyczną transformacją struktur Kripke, rozumianą jako sekwencja operacji obejmujących dodawanie i usuwanie etykietowań w strukturach Kripke, to możemy sformułować definicje o zachowaniu spełnialności, a także inne pojęcia.

Definicja 1 (Zachowanie spełnialności [A5]) Niech Fr_1 będzie zbiorem poprawnie utworzonych formuł. Zbiór jest *spełniony* wtedy i tylko wtedy gdy $\mathsf{K}_1 \models Cn(Fr_1)$, gdzie K_1 jest strukturą Kripke, i Cn koniunkcją wszystkich formuł. Transformacja τ, dla $Fr_2 = \tau(Fr_1)$, *zachowuje spełnialność* wtedy i tylko wtedy gdy Fr_2 jest także spełniony, to

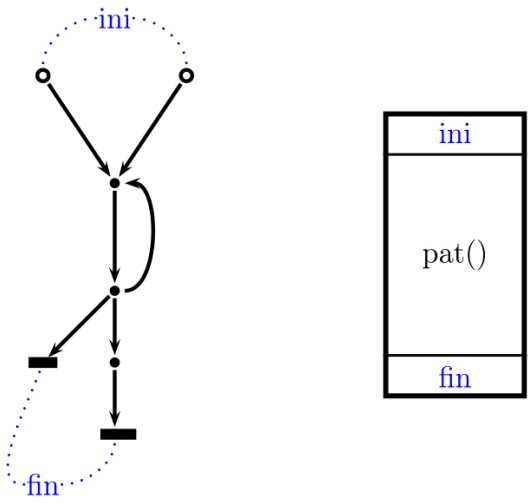


Rysunek 7: Ilustracja kompozycyjności i sposób korzystania z niej [A5], punkty z lewej strony oznaczają argumenty atomowe wyrażenia wejściowego w, ramki oznaczają zagnieżdżanie użytych wzorców, strzałki oznaczają użycie argumentów w odniesieniu do zbioru Σ, celem wygenerowania pewnej sub-specyfikacji, podwójne strzałki oznaczają operację konsolidowania wyrażeń, linie pionowe oznaczają sub-specyfikację użytą do wytworzenia fragmentu innej i większej specyfikacji, a całość operacji jest powtarzana aż do zbudowania wynikowej specyfikacji logicznej L

znaczy $\mathsf{K}_2 \models Cn(Fr_2)$, gdzie K_2 jest strukturą Kripke. Wynikowe formuły charakteryzuje się jako *zachowujące spełnialność*.

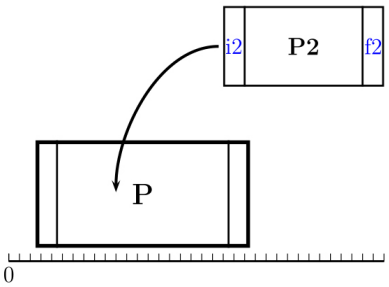
W tabeli 1 zostały pokazane, jako przykładowe, zdefiniowane wzorce logiczne, zwane również prymitywami logicznymi, por. także rysunek 8. Każdy wzorzec logiczny, dokładna definicja wzorca w pracach [A4, A5], składa się po pierwsze z dwóch formuł opisujących w logice klasycznej warunki inicjujące oraz finalizujące wzorzec, odpowiednio *ini* oraz fin, a resztę formuł stanowią formuły PLTL opisujące własności logiczne wzorca, jego zachowanie, tj. zależności logiczne wewnętrznych zadań, w terminach logiki formalnej. Formuły takie opisują zawsze aspekty żywotnościowe (ang. *liveness*) oraz bezpieczeństwa (ang. *safety*), por. [A5, B11]. Jednak wzorce logiczne muszą być jeszcze właściwie, z punktu widzenia metody ΠBCD, sformułowane.

Definicja 2 (Poprawnie skomponowany wzorzec [A5]) Wzorzec logiczny P jest *logicznie poprawnie skomponowany*, wtedy i tylko wtedy, gdy wszystkie elementarne formuły $P.pat()$ są spełnione i niesprzeczne, oraz $P.pat \models P.Tr$, gdzie $P.Tr$ są *formułami przejścia* dla pewnego wzorca P, i $P.Tr \equiv Tr \equiv \{\Box\neg(P.ini \wedge P.fin), \Diamond P.ini, \Box(P.ini \Rightarrow \Diamond P.fin)\}$.



Rysunek 8: Ilustracja wzorców [A5]. Po lewej – wzorzec behawioralny z jego zadaniami oraz warunkami inicjującymi i finalizującymi, po prawej – wzorzec logiczny ze zbiorem formuł $pat()$ opisującym jego własności logiczne (PLTL) oraz wrunki inicjujące i finalizujące (klasyczny rachunek zdań)

Formuły przejścia również precyzują zarówno własności bezpieczeństwa jak i żywotności poprawnie sformułowanego wzorca.



Rysunek 9: Wstrzykiwanie wzorca do wzorca [A5] ($i2$ oraz $f2$ oznaczają logiczne warunki początkowe oraz końcowe, które opisują sytuację rozpoczęcia/zakończenia wzorca i jednocześnie reprezentują go na zewnątrz.)

Zagnieżdżanie wzorców w wyrażeniu logicznym powoduje jakby wstrzyknięcie wzorców wraz z ich formułami logicznymi do innego wzorca, por. rysunek 9. Na bazie tych pojęć, ale i wcześniejszych wyników, w pracy [A5], sformułowano i udowodniono twierdzenie o *zachowaniu spełnialności* przez algorytm ΠC. Jest to jedno z kluczowych twierdzeń, gdyż gwarantuje, że w wynikowa specyfikacja logiczna, wygenerowana przez zaproponowany w cyklu prac algorytm, jest spełniona, o ile zaprojektowany model zachowania był spełniony, i odwrotnie. Tak więc, mówiąc nieformalnie, wygenerowana specyfikacja logiczna nie pogorszy się w sensie spełnialności. Z kolei w pracy [A4] sformułowano twierdzenie

o *względnej zupełności* algorytmu. Jest to rozumiane w ten sposób, mówiąc nieformalnie, że wygenerowana specyfikacja logiczna nie odejmuje niczego z pierwotnej, tj. względem pierwotnej, specyfikacji modelu zachowania. Tak więc, wygenerowana specyfikacja logiczna nie pogorszy się w znaczeniu zupełności.

Tabela 2: Metody systemów wnioskujących [A5]

Formuły	Uwagi
$Cn(L)$, $Rq \wedge Cn(L)$	spełnialność, niespełnialność, podstawowe własności logiczne specyfikacji, otwarte i zamknięte gałęzie w drzewie prawdy, sprzeczność
$Cn(L) \Rightarrow Rq$, $\neg(Cn(L) \Rightarrow Rq)$	własności logiczne które są konsekwencją, od przesłanek do konkluzji, logiczne wynikanie, tautologiczność, twierdzenie o wnioskowaniu, *reductio ad absurdum*
$\frac{Cn(L) \Rightarrow Rq, Cn(L)}{Rq}$	*modus ponens*

Specyfikacja logiczna dla każdego modelu zachowania pokazuje formalnie strukturę i dynamikę działania systemu. W związku z tym, uzyskana specyfikacja logiczna modelu zachowania, wygenerowana automatycznie w oparciu o algorytm ΠC, daje duże możliwości jej analizy, a więc także praktycznego wykorzystania wyników wchodzących w przedstawiony cykl prac, por. rysunek 4. W pracach [A4, A5] pokazano liczne przykłady wygenerowanych zbiorów formuł. Oczywiście dla rzeczywistych, o skali przemysłowej, modeli zachowania, zbiory takie będą o wiele bardziej liczebne, składając się z wielu setek formuł. W planowanym środowisku F-IDE może być analizowanych, po wbudowaniu systemu automatycznego dowodzenia twierdzeń, wiele logicznych własności: spełnialność, niespełnialność, spójność logiczna, tautologiczność, i szereg innych. Możliwe jest także badanie spójności horyzontalnej (modele zachowania na tym samym poziomie projektowym), lub spójności wertykalnej (modele zachowania dla różnych wersji tego samego artefaktu). Procesy wnioskowania mogą być przeprowadzone w oparciu o różne metody, por. tabela 2, co po części jest uzależnione od dostępnych silników wnioskujących.

Tabela 3: Metody analizy kontekstu zachowania [A5]

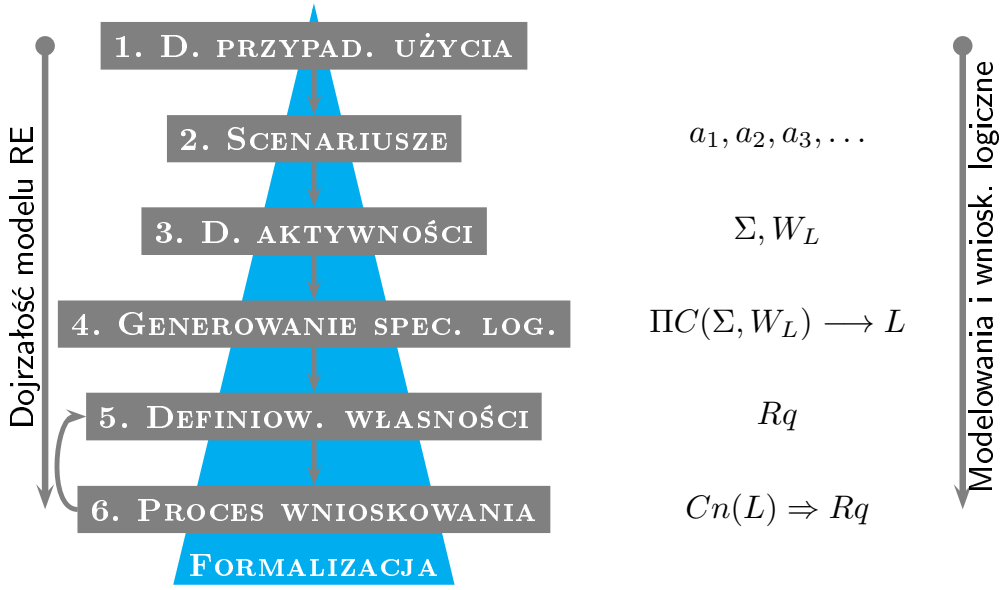
Sformułowanie	Znaczenie
kontradyktoryjność	logiczna opozycja, uzyskane specyfikacje nie mogą być spełnione jednocześnie i nie mogą być niespełnione jednocześnie
przeciwstawność	wzajemna opozycja, uzyskane specyfikacje nie mogą być spełnione jednocześnie, ale mogą być niespełnione jednocześnie
podprzeciwstawność	uzyskane specyfikacje nie mogą być niespełnione jednocześnie, ale mogą być spełnione jednocześnie
podporządkowanie	pierwsza specyfikacja pociąga logicznie drugą, ale nie odwrotnie

Jeszcze inne, ale bogate możliwości logiczne, stwarza wnioskowanie w oparciu o tzw.

kwadrat logiczny, por. tabela 3, co pozwala na przykład na przebadanie odmiennych i alternatywnych scenariuszy zachowania danego modelu, a także całego kontekstu logicznego.

Inżynieria wymagań RE ([A2, A3, A5]) Przedstawione wyniki cyklu prac mogą być z powodzeniem wykorzystane do analizy i weryfikacji modeli oprogramowania w inżynierii wymagań RE. Powodzenie fazy RE ma fundamentalne znaczenie dla powodzenia całego projektu [C40, C41]. Faza RE dla systemów o podwyższonych wymaganiach bezpieczeństwa (ang. *safety critical systems*) ma znaczenie krytyczne. Uzyskane z różnych źródeł artefakty, służące do budowania modeli zachowania, po transformacji ich do postaci logicznej specyfikacji, mogą być wykorzystane do badania co najmniej logicznej spójności, a także innych relacji logicznych, dostarczając tym samym bardzo cennych wskazówek, jak rozwiązać ewentualnie pojawiające się logiczne problemy i niespójności [C31].

W cyklu prac została zaproponowana metoda budowania, rozwijania i weryfikowania modeli przeznaczonych dla fazy RE, modeli zbudowanych na bazie diagramów języka UML. Metoda w końcowej fazie wykorzystuje opracowany algorytm ΠC. Celem metody jest przejście, krok za krokiem, od artefaktów wyrażonych narracyjnie do formalnej specyfikacji logicznej. Pierwsze próby w odniesieniu do metody zawarte są w pracy [B12], później także [B13, A2], a ostatnia wersja metody w pracy [A3]. Na rysunku 10 został pokazany ogólny schemat postępowania dla modeli RE.



Rysunek 10: Kolejne kroki dla RE w proponowanej metodzie, por. [A3]

Zaproponowaną w pracach cyklu procedurę można sformułować jako sekwencję następujących kroków, por. także rysunek 11 z transformacjami modeli:

1. Utworzenie diagramów przypadków użycia $UCD_{d=1,\ldots}$ – modelowanie i rozumienie funkcji i serwisów systemu.

2. Modelowanie poszczególnych przypadków użycia $UC_{i=1,...}$ – rozumienie pojedynczych usług i serwisów.

3. Modelowanie scenariuszy przypadków użycia w postaci tekstu strukturalizowanego, identyfikowanie aktywności atomowych $\{a_1, a_2, a_3, ...\}$ oraz modelowanie scenariuszy alternatywnych.

4. Modelowanie diagramów aktywności $AD_{i=1,...}$ dla każdego scenariusza, modelowanie sieci działań z użyciem zidentyfikowanych aktywności atomowych $\{a_1, a_2, a_3, ...\}$ oraz predefiniowanych wzorców behawioralnych Π.

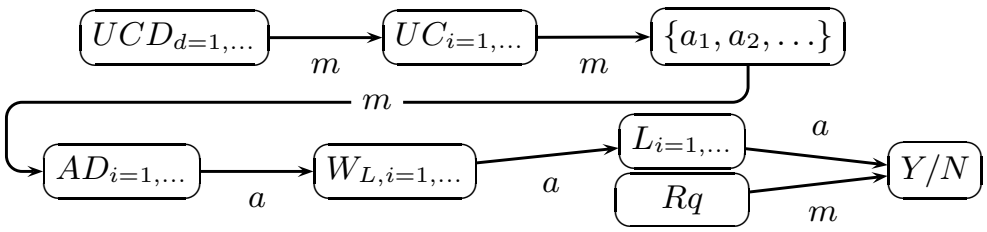
5. Automatyczne konwertowanie diagramów aktywności do wyrażeń logicznych $W_{L,i=1,...}$

6. Automatyczne generowanie specyfikacji logicznej dla każdego wyrażenia logicznego, $\Pi C(W_{L,j=1,...}, \Sigma) \longrightarrow L_{j=1,...}$, tzn. translacja każdego wyrażenia logicznego z $W_{L,j}$ do specyfikacji logicznej L_j, rozumianej jako zbiór formuł PLTL.

7. Definiowanie żądanych własności modelu zachowania, poprzez wprowadzanie formuł dodatkowych, poprawnościowych, np. Rq, ale także realizowanie innych scenariuszy, por. [A5, podrozdział 6.4].

8. Formalna weryfikacja żądanych własności w środowisku F-IDE, por. rysunek 4, a także tabele 2 i 3.

9. Z ostatniego kroku można powrócić do dowolnego wcześniejszego kroku, aby dokonać modyfikacji modelu zachowania i następnie ponownie przeprowadzić weryfikację logiczną modelu. Takie postępowanie może być powtarzane wielokrotnie i stosownie do potrzeb.



Rysunek 11: Transformacje modeli w zaproponowanej metodzie RE. "m" oznacza przejście manualne lub pół-manualne, "a" oznacza przejście automatyczne.

Procesy weryfikacji i wnioskowania dedukcyjnego mogą być dowolnie rozbudowane, co wynika z potrzeb konkretnego analityka i projektanta, a także możliwości zaprojektowanego środowiska F-IDE. Przedstawione poniżej scenariusze [A5] dają dobry przegląd możliwości.

— Przypuśćmy, że wygenerowana specyfikacja logiczna dla sieci działań lub diagramu aktywności, jest następująca: $L = \{\Diamond a, \Box(a \Rightarrow \Diamond b), \Box(b \Rightarrow \Diamond c), \Box\neg(a \wedge b), \Box\neg(b \wedge c)\}$. W rzeczywistości, typowa specyfikacja logiczna może zawierać kilkadziesiąt, lub znacznie więcej, formuł wraz z dziesiątkami zmiennych. Z kolei wymaganiami

poprawnościowymi mogą być zdania: $Rq_1 \equiv \Box(a \Rightarrow \Diamond c)$ oraz $Rq_2 \equiv \Box\neg(a \wedge b \wedge c)$. Wówczas, badane formuły to $Cn(L) \Rightarrow Rq_1$ dla własności żywotności oraz $Cn(L) \Rightarrow Rq_2$ dla własności bezpieczeństwa.

— Następnie przypuśćmy, że mamy kilka alternatywnych diagramów aktywności A_1, A_2, $A_3, \ldots$, będących sieciami działań, lub po prostu ich równoważniki w postaci wyrażeń logicznych. Są one alternatywnymi scenariuszami pewnego przypadu użycia UML, dla których wygenerowano specyfikacje logiczne $L(A_1), L(A_2), L(A_3), \ldots$ Wówczas możemy wymagać, aby pewna własność Rq była spełniona dla każdego scenariusza, przy rozważaniu ich łącznie, jako własność podstawowa i wspólna, to znaczy $Cn(L(A_1)) \wedge Cn(L(A_2)) \wedge Cn(L(A_3)) \models Rq$.

— Ponadto, może być koniecznym przygotowanie dwóch diagramów aktywności, powiedzmy A oraz A', jako przeciwnych scenariuszy pewnego przypadku użycia, co w efekcie oznacza, że obie specyfikacje logiczne są kontradyktoryjne, a mianowicie jest spełnione $Cn(L(A)) \Rightarrow \neg Cn(L(A'))$, ale także jest spełnione $\neg Cn(L(A)) \Rightarrow Cn(L(A'))$.

— Następnie, dla pewnych diagramów aktywności, powiedzmy A oraz A', pewnego przypadku użycia, nie może być tak, że obie wygenerowane specyfikacje są niespełnione, a więc obie specyfikacje są podprzeciwstawne, to znaczy, że jest spełnione $\neg(\neg Cn(L(A)) \wedge \neg Cn(L(A')))$.

— Wreszcie, dla dwóch scenariuszy pewnego przypadku użycia, pewien diagram aktwności A jest podstawą do wyprowadzenia innego diagramu A', który jest mu podporządkowany (jako akcja wywołania działania), to znaczy, ich logiczne specyfikacje są podporządkowane w ten sposób, że jest spełnione $Cn(L(A)) \Rightarrow Cn(L(A'))$, ale także jest spełnione $\neg(Cn(L(A')) \Rightarrow Cn(L(A)))$.

— Poza powyższym przypadkami, należy także testować i monitorować logiczne relacje pomiędzy różnymi diagramami aktywności w odniesieniu do wertykalnej i horyzontalnej spójności logicznej.

Przedstawione przykłady scenariuszy otwierają także nowe kierunki badawcze, szczególnie w kontekście F-IDE, zarówno w odniesieniu do organizacji samego narzędzia F-IDE, planowanych do wbudowania silników wnioskujących, realizowanych scenariuszy dowodzenia, jak i ogólnie zastosowania języka UML w procesach inżynierii wymagań.

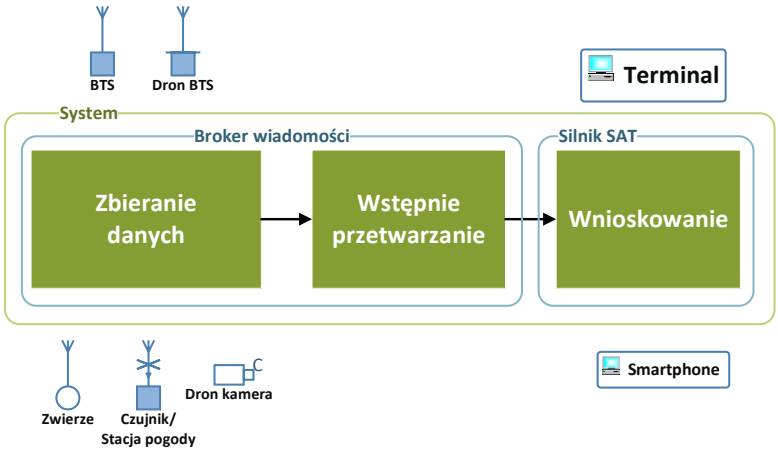
Inteligentne środowiska IE ([A6, A7, A8, A9, A10]) Problematyka inteligentnego miasta (ang. *smart city*), albo szerzej inteligentnych środowisk IE (ang. *Inteligent Environments*), od początku była w kręgu zainteresowań habilitanta. Wpisuje się ona także w takie obszary badawcze jak inteligencja otoczenia AmI (ang. *Ambient Intelligence*), Internet rzeczy IoT (ang. *Internet of Things*), czy ogólniej, obliczenia rozpowszechnione i wszechobecne (ang. *pervasive and ubiquitous computing*). Celem ich jest dostarczenie systemów wspierających działalność i życie codzienne człowieka, zwiększających komfort życia mieszkańców, wpływających na podwyższenie ich bezpieczeństwa i jakość oferowanych usług, por. manifest [C42], czy [C43]. Aby osiągnąć zamierzone cele, systemy takie analizują olbrzymie wolumeny różnorodnych danych kontekstowych, por. prace [C44, C45], tj.

danych wytworzonych w otoczeniu mieszkańca, z jego udziałem lub bez. Przeprowadzane są różne i bardzo złożone procesy wnioskowania, a zasadniczym celem tych aktywności jest dostarczenie w sposób pro-aktywny, transparetny dla mieszkańców, ale i świadomy kontekstu, konkretnych usług i działań wpisanych w wymienione wcześniej założenia. Wnioskowanie logiczne, wspierajace w tych obszarach procesy decyzjne, jest naturalnym komponentem takich systemów. Automatyzacja procesów wnioskowania logicznego, m.in. poprzez nowe rozwiązania w zakresie silników SAT, akceleruje kolejne etapy działania inteligentnych systemów.

Powstały liczne prace dotyczące inteligentnego miasta i środowiska. W pracy [A6] zostały zaproponowane rozwiązania dla systemu zarządzania oświetleniem ulicznym. Przedstawiono wymagania takiego systemu, jego podstawową architekturę, a system pomyślano jako świadomy kontekstu i reagujący na jego zmiany. W procesach wnioskowania zastosowano znany język modelowania kontekstu CML (ang. *Context-Modeling Language*), który jednak został w pracy rozszerzony, gdyż w wersji podstawowej jest niewystarczający w modelowaniu logicznym złożonych sytuacji. Specyfikacja logiczna, w odniesieniu do której przeprowadza się wnioskowanie, jest wyrażona w logice temporalnej PLTL. W pracy [A7] został zaproponowany trzy-warstwowy system wielo-agentowy, wraz z wnioskowaniem w logice PLTL tak, aby rejestrować preferencje użytkowników korzystających z parkingu miejskiego. W oparciu o zgromadzone informacje proponowane są najlepsze dla użytkownika miejsca parkingowe. Praca [B14] pokazuje rozwiązania dla inteligentnego, publicznego transporu miejskiego, został zaproponowany system wielo-agentowy, a po raz pierwszy w pracach zasugerowano pobieranie danych bezpośredno ze stacji BTS (ang. *Base Transiver Station*), w postaci rekordów CDR. Wnioskowanie odbywa się poprzez dynamicznie tworzony i analizowany graf podróży. Z kolei praca [B15] proponuje kompleksowe rozwiązania dla inteligentnego domu. Inteligentne scenariusze umożliwiają przełączanie urządzeń w pożądany sposób, gdy określona aktywność użytkownika uważana za wyzwalacz jest spełniona. Rozszerzeniem pracy [B14] jest praca [B16], gdzie dane ze stacji BTS zostały uwzględnione do zarządzania inteligentnym oświetleniem ulicznym. Praca [B17] również dotyczy oświetlenia ulicznego. Zostały zaproponowane nowe scenariusze działania oraz system wielo-agentowy. Z kolei praca [B18] proponuje system do inteligentnego zarządzania kolejkami klientów tworzącymi się w obiektach użyteczności publicznej. Klienci są identyfikowani i rozpoznawani oraz w zależności od ich statusu i bieżącej sytuacji kierowani do właściwych kolejek. Odmienną klasę systemu stanowi koncepcja inteligentnego pistoletu, który miałby być na wyposażeniu policji [B19]. Urządzenie to, oprócz normalnych funkcji strzelniczych, uczestniczy w organizowaniu wsparcia dla interweniującego policjanta, a także powiadamianiu odpowiednich służb miejskich. Z kolei w pracy [B20] analizowane były dane pobierane z mediów społecznościowych, celem budowania modelu preferencji użytkownika. Wnioskowanie jest przeprowadzane w oparciu o silniki SAT. Wreszcie w pracy [A8] są rozważane duże zbiory danych dla których zostały zaproponowane zorientowane logicznie narzędzia oraz algorytmy, służące do identyfikowania wzorców logicznych i budowania dla nich specyfikacji logicznej, która następnie wykorzystywana jest w systemach wnioskowania. Została zaproponowana architektura takiego systemu oraz przykłady wnioskowania.

Jednakże najbardziej zaawansowana wersja systemu dla inteligentnego środowiska została opracowana dla systemu wspierającego pracę ratowników górskich. Najpierw była to praca [B21], a później kulminacja w pracy [A9], por. także rysunek 12. Celem pracy

było pokazanie, że bezpośrednie sparowanie dwóch technologii, brokerów przetwarzających duże strumienie danych kontekstowych oraz silników SAT, może dać dobre efekty w postaci systemu klasy AmI, operującego w inteligentnym środowisku IE, a ponadto działającego w koncepcji obliczeń rozpowszechnionych i wszechobecnych. Został zaproponowany model danych kontekstowych, por. rysunek [A9, rys. 1], który jest adekwatny dla środowiska górskiego oraz monitorowanych obiektów. System na bazie wnioskowania logicznego dokonuje kwalifikacji sytuacji kontekstowej dla każdej osoby znajdującej się w monitorowanym obszarze, poprzez przypisanie do tej osoby jednego z pięciu poziomów zagrożenia, rosnąco od E1 do E5. (Wizualnie odbywa się to poprzez przypisanie do każdej osoby odpowiedniej kolorowej ikonki.) Przypisania mogą się zmieniać w czasie przebywania w rejonie, w miarę zmieniających się warunków atmosferycznych, także w miarę przemieszczania się w monitorowanym obszarze i zmiany tras do których są przypisane różne poziomy trudności. Alerty o zagrożeniu mogą także wynikać z pojawiania się innych nietypowych zagrożeń, takich jak niebezpieczne zwierzęta, odłączenie się od grupy, zejście z wyznaczonych szlaków górskich, niepokojąco długiego postoju na szlaku, tj. poza miejscami naturalnymi dla takiego postoju, jak np. schroniska, itd. Ratownicy dzięki zaprojektowanemu systemowi mogą łatwo obserwować sytuację w całym monitorowanym obszarze, w odniesieniu do każdej osoby, i stosownie do tego podejmować swoje działania (ratownicze), o ile są konieczne. Ratownicy mogą też, w razie potrzeby, przedefiniowywać poziomy reagowania, które są podane w postaci tabeli, por. [A9, tabela 5], w formie zmiennych logicznych, obejmujących: zdefiniowane poziomy zagrożenia dla turysty, poziomy trudności dla poszczególnych szlaków, wielkości poszczególnych parametów związanych z warunkami atmosferycznymi – wszystko podane w postaci ograniczeń logicznych.

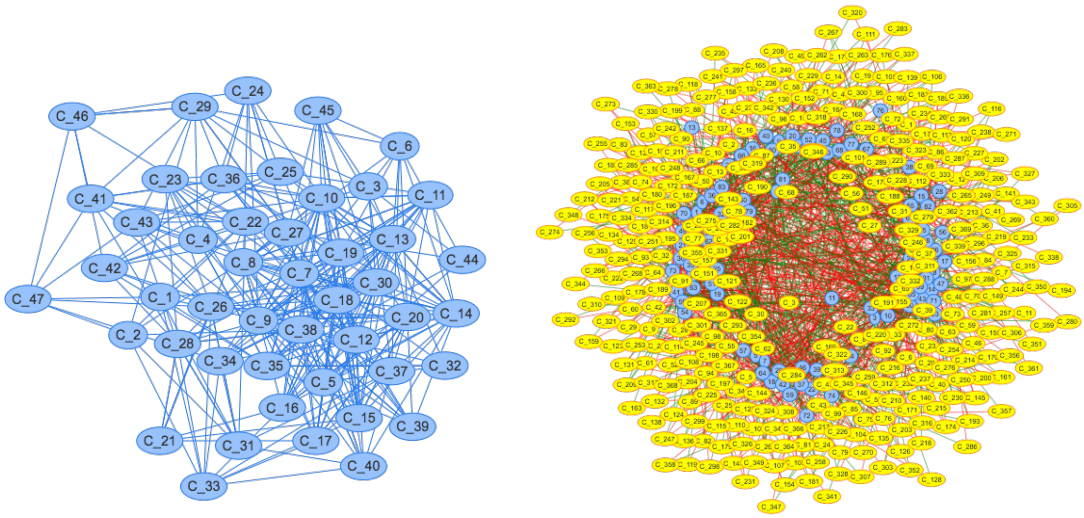


Rysunek 12: Ogólny schemat systemu wykrywania zagrożeń, wspierającego pracę ratowników górskich [A9]

System przetwarza duże ilości danych składających się na kontekst, pochodzących zarówno z licznie rozlokowanych w monitorowanym obszarze stacji meteo, dostarczających podstawowe informacje pogodowe w tym zakresie (temperatura, deszcz, wiatr, mgła) oraz danych geolokalizacyjnych dotyczących poszczególnych turystów, a pochodzących ze znajdujących się w monitorowanym obszarze stacji telefonii komórkowej BTS. Wolumen

tak uzyskanych danych jest powiększany jeszcze o informacje o zagrożeniach lawinowych, a także o innych zagrożeniach. System podejmuje decyzje na bazie wnioskowania logicznego. Po wstępnym przetworzeniu wszystkich danych (preprocessing), które są najpierw filtrowane, zagregowane i przeformatowywane tak, aby umożliwić wnioskowanie dedukcyjne, wchodzą one w bezpośrednią interakcję z silnikami SAT. Dane w całym systemie są transportowane przez brokery wiadomości, a wnioskowanie odbywa się z wykorzystaniem wybranego silnika SAT.

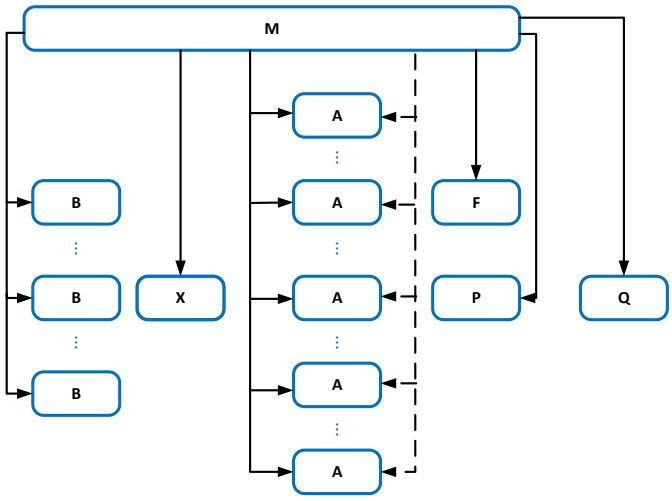
Aby osiągnąć zamierzone cele, w pracy [A9] opracowano koncepcję funkcjonalną systemu, a także jego architekturę. Zaprojektowano wszystkie podstawowe algorytmy systemu. Zostały one przeanalizowane pod względem poprawności. Zbadano także złożoności obliczeniowe wszystkich algorytmów oraz przeprowadzono liczne testy, obejmujące m.in. transport danych. Testowano przy tym dwa podstawowe brokery, a także konfigurację chmurowową. Testy dostarczyły ciekawe wyniki charakteryzujące przebieg procesów transportowych. Przeprowadzono testy kilkunastu silników SAT. Materiał testowy, jak i uzyskane wyniki są bardzo bogate, bowiem obejmują nie tylko podstawowe parametry czasowe, ale także indywidualne charakterystyki wielu silników. Należy stwierdzić, że przeprowadzone w pracy [A9] badania teoretyczne i empiryczne udowodniły wykonalność systemu, jego żywotność, osiąganie celów w zamierzonym krótkim czasie. System reaguje na zdarzenia w czasie rzeczywistym i on-line. Wnioskowanie logiczne i podejmowanie decyzji w oparciu o silniki SAT okazało się bardzo wygodne i skuteczne w inteligentnych systemach klasy IE.



Rysunek 13: Przykładowe wizualizacje formuł logicznych pochodzące z ogólnodostępnego, własnego serwisu http://forvis.agh.edu.pl

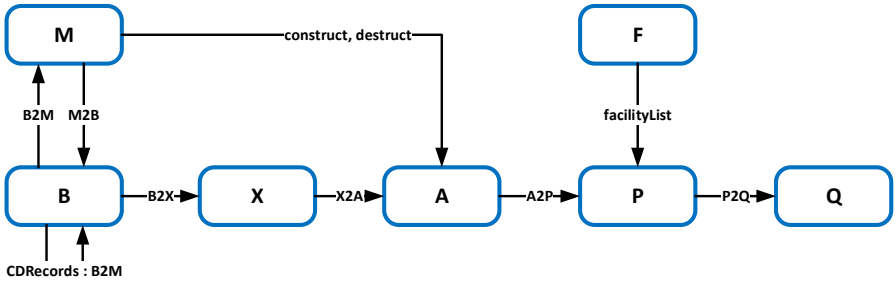
Z formułami logicznymi, szczególnie dużych rozmiarów, wiąże się znany problem ich wizualizacji. W pracy [B22] został opisany niedawno stworzony i ogólnodostępny serwis, wówczas jeszcze działajacy tylko w oparciu o znane w literaturze metody wizualizacji, por. rysunek 13. Obecnie dodano nowe metody, w tym obejmujące problem ważonego MaxSat. Wizualizacja nie jest tylko zagadnieniem estetycznym, ale może także dostarczać ważnych informacji o strukturze wewnętrznej zakodowanego problemu, a mówiąc niefor-

malnie, o jego regularności, albo przeciwnie, o istnieniu tzw. *backbones* czy *backdoors*, których wykrycie, jak wiadomo, znacznie przyśpiesza działanie silników SAT. Planowany jest dalszy rozwój systemu, a zainicjowany serwis jest początkiem kolejnych prac. Serwis w zamierzeniu ma gromadzić zainteresowanych problematyką badaczy, jak i zwykłych użytkowników. Planowany jest także kolejny serwis, który będzie pozwalał na losowe generowanie formuł logicznych dla klasycznego problemu SAT, z możliwością bogatego dobierania różnych parametrów dla sterowania procesem generowania, por. [B23].



Rysunek 14: Architektura proponowanego systemu agentowego, linie ciągłe – powołanie do życia agenta, linie przerywane – destrukcja agenta [A10]

Systemy wielo-agentowe są obecnie w większości omawianych tutaj rozwiązań, autorstwa lub współautorstwa habilitanta, dla systemów klasy IE. Są obecnie także w kolejnym wątku, jakim jest metoda konstruowania trajektorii przemieszczania się obiektów w monitorowanym obszarze miejskim, opisana wstępnie w pracy [B21], a już dokładniej zaproponowana w pracy [A10]. Przetwarzane dane pochodzą ze stacji telefonii komórkowej BTS, w postaci rekordów CDR. Został zaproponowany system wielo-agentowy, por. rysunek 14, generujący indywidualne trajektorie przemieszczania się obiektów. Trajektorie są następnie wzbogacane o informacje o wybranych elementach infrastruktury miejskiej, na które natrafił monitorowany obiekt, por. także rysunek 15.



Rysunek 15: Przepływy strumieni danych pomiędzy agentami [A10]

Tak uzyskane trajektorie mogą być następnie poddane odrębnej analizie, celem stworzenia pogłębionej charakterystyki przemieszczania się i pobytu badanego obiektu w monitorowanym obszarze. Ciekawym i praktycznym obszarem zastosowania tej koncepcji jest profilowanie zachowania się zwiedzających, poruszających się w określonym ośrodku turystycznym. Planuje się także wykorzystanie do tego celu wnioskowania logicznego.

Podsumowanie Na rysunku 3 został pokazany zakres prac przedstawionego cyklu, a także jego główne obszary badawcze. W obszarze IO projektowanie i wytwarzanie poprawnego i wiarygodnego oprogramowania, w związku z rosnącymi oczekiwaniami, potencjalnie dużymi kosztami błędnych wykonań, nieprzewidywalnością działania oraz trudnościami usuwania zbyt późno wykrytych błędów, ma fundamentalne znaczenie poprzez eliminowanie błędów strukturalnych modeli behawioralnych już w fazie RE. W obszarze IE automatyczne modelowanie procesów decyzyjnych, pokazane na wybranych przykładach, może być z powodzeniem szeroko stosowane i wdrażane w innych zastosowaniach inteligentnego środowiska. Oba zaproponowane podejścia mają więc duże znaczenie aplikacyjne.

Podstawowym elementem oryginalnego wkładu naukowego zawartego w przedstawionych pracach jest zaproponowanie metody, która daje formalne podstawy do zbudowania środowiska F-IDE, z przeznaczeniem do zastosowana w procesach projektowania i modelowania inżynierii oprogramowania. Wszystkie elementy wkładu można wyliczyć następująco:

— metoda IIBCD, i jej formalne podstawy, pozwalająca na automatyczne generowanie specyfikacji logicznej modeli zachowania opartych na sieciach działań,

— zdefiniowanie w terminach struktury Kripke, powiązanej z przedziałami całkowitoliczbowymi, wzorców behawioralnch, oraz w terminach logiki temporalnej PLTL kilkunastu wzorców logicznych (logicznych prymitywów),

— zaproponowanie pośredniej reprezentacji tekstowej dla modeli behawioralnych opartych na sieciach działań, zaproponowanie algorytmu wzynaczania wyrażenia skonsolidowanego dla zagnieżdżonych wyrażeń logicznych,

— zaproponowanie podstawowego algorytmu IIC generowania specyfikacji logicznej w logice temporalnej PLTL,

— udowodnienie poprawności działania algorytmu generowania, oraz faktu, że rozpina on drzewo dla każdego wyrażenia wejściowego algorytmu, oraz wykazanie jego złożoności obliczeniowej,

— udowodnienie zachowania spełnialność dla uzyskiwanej specyfikacji logicznej, oraz względnej zupełności specyfikacji,

— pokazanie, że metoda IIBCD jest silnie zorientowana kompozycyjnie, co pozwala zarządzać złożonością w przypadku rozbudowanych modeli behawioralnych oprogramowania,

— pokazanie licznych scenariuszy procesów wnioskowania logicznego, z wykorzystaniem wygenerowanej specyfikacji logicznej, planowanych dla przyszłego narzędzia klasy F-IDE,

— zaproponowanie metody budowania modeli behawioralnych dla procesów inżynierii wymagań RE, bazujących na diagramach języka UML, która będzie wykorzystana w przyszłym narzędziu F-IDE,

— zbudowanie koncepcji wielu systemów klasy IE/AmI/IoT na bazie systemów wielo-agentowych oraz uwzględniajacych wnioskowanie, i podejmowanie decyzji, w oparciu o silniki SAT,

— zaproponowanie zaawansowanego systemu klasy IE wspierającego służby ratownicze, w którym sparowano zaawansowane borkery wiadomości oraz silniki SAT,

— udowodnienie poprawności formalnej zaproponowanego systemu oraz przeprowadzenie licznych eksperymentów wykazujących wykonalność i efektywność przetwarzania złożonych danych kontekstowych,

— utworzenie i zweryfikowanie koncepcji systemu wielo-agentowego do budowania indywidualnych trajektorii przemieszczania się obiektów na terenie miejskim w oparciu o dane ze stacji BTS.

Literatura

Prace stanowiące cykl publikacji (≡ rozdz. 4.2)

[A1] R. Klimek, "Towards formal and deduction-based analysis of business models for SOA processes," in *Proceedings of 4th International Conference on Agents and Artificial Intelligence (ICAART 2012), 6–8 February, 2012, Vilamoura, Algarve, Portugal*, J. Filipe and A. Fred, Eds., vol. 2. SciTePress, 2012, pp. 325–330.

[A2] ——, "Deduction-based formal verification of requirements models with automatic generation of logical specifications," in *7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012, Revised Selected Papers), 29–30 June, 2012, Wrocław, Poland*, ser. Communications in Computer and Information Science, L. Maciaszek and J. Filipe, Eds., vol. 410. Springer-Verlag, 2013, pp. 157–171.

[A3] ——, "From extraction of logical specifications to deduction-based formal verification of requirements models," in *Proceedings of 11th International Conference on Software Engineering and Formal Methods (SEFM 2013), 25–27 September 2013, Madrid, Spain*, ser. Lecture Notes in Computer Science, R. M. Hierons, M. G. Merayo, and M. Bravetti, Eds., vol. 8137. Springer Verlag, 2013, pp. 61–75.

[A4] ——, "A system for deduction-based formal verification of workflow-oriented software models," *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 4, pp. 941–956, 2014. [Online]. Available: http://www.amcs.uz.zgora.pl/?action=paper&paper=802

[A5] ——, "Pattern-based and composition-driven automatic generation of logical specifications for workflow-oriented software models," *Journal of Logical and Algebraic Methods in Programming*, vol. 104, pp. 201–226, 2019.

[A6] R. Klimek and G. Rogus, "Modeling context-aware and agent-ready systems for the outdoor smart lighting," in *Proceedings of 13th International Conference on Artificial Inteligence and Soft Computing (ICAISC 2014), 1–5 June, 2014, Zakopane, Poland*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., vol. 8468. Springer Verlag, 2014, pp. 269–280.

[A7] R. Klimek and L. Kotulski, "Proposal of a multiagent-based smart environment for the IoT," in *Workshop Proceedings of the 10th International Conference on Intelligent Environments, Shanghai, China, 30th June–1st of July 2014*, ser. Ambient Intelligence and Smart Environments, J. C. Augusto and T. Zhang, Eds., vol. 18. IOS Press, 2014, pp. 37–44.

[A8] R. Klimek, "Behaviour recognition and analysis in smart environments for context-aware applications," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC 2015), October 9–12, 2015, City University of Hong Kong, Hong Kong*. IEEE Computer Society, 2015, pp. 1949–1955.

[A9] ——, "Exploration of human activities using message streaming brokers and automated logical reasoning for ambient-assisted services," *IEEE Access*, vol. 6, pp. 27 127–27 155, 2018.

[A10] ——, "Towards recognising individual behaviours from pervasive mobile datasets in urban spaces," *Sustainability*, vol. 11, no. 6, 2019.

Prace habilitanta nie wchodzące w skład cyklu

[B11] R. Klimek, *Wprowadzenie do logiki temporalnej*. Wydawnictwa Naukowo-Techniczne AGH, 1999.

[B12] R. Klimek and P. Szwed, "Formal analysis of use case diagrams," *Computer Science*, vol. 11, pp. 115–131, 2010.

[B13] R. Klimek, "Proposal to improve the requirements process through formal verification using deductive approach," in *Proceedings of 7th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2012), 29–30 June, 2012, Wrocław, Poland*, J. Filipe and L. Maciaszek, Eds. SciTePress, 2012, pp. 105–114.

[B14] R. Klimek and L. Kotulski, "Towards a better understanding and behavior recognition of inhabitants in smart cities. a public transport case," in *Proceedings of 14th International Conference on Arificial Inteligence and Soft Computing (ICAISC*

2015), 14–18 June, 2015, Zakopane, Poland, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., vol. 9120. Springer Verlag, 2015, pp. 237–246.

[B15] R. Klimek and G. Rogus, "Proposal of a context-aware smart home ecosystem," in *Proceedings of 14th International Conference on Arificial Inteligence and Soft Computing (ICAISC 2015), 14–18 June, 2015, Zakopane, Poland*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., vol. 9120. Springer Verlag, 2015, pp. 412–423.

[B16] R. Klimek, L. Kotulski, and A. Sędziwy, "Behavioural patterns from cellular data streams and outdoor lighting as strong allies for smart urban ecosystems," in *State of the Art in AI Applied to Ambient Intelligence. Proceedings of the 10th Anniversary Edition Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI 2015), International Joint Conference on Artificial Intelligence (IJCAI 2015), 25–27 July 2015, Buenos Aires, Argentina*, ser. Frontiers in Artificial Intelligence and Applications, A. Aztiria, J. C. Augusto, and A. Orlandini, Eds., vol. 298. IOS Press, 2017, pp. 109–121.

[B17] R. Klimek, "Proposal of a multi-agent system for a smart outdoor lighting environment," in *Proceedings of 16th International Conference on Arificial Inteligence and Soft Computing (ICAISC 2017), 11–15 June, 2017, Zakopane, Poland*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., vol. 10246. Springer International Publishing, 2017, pp. 255–266.

[B18] ——, "Context-aware and pro-active queue management systems in intelligent environments," in *Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS 2017), 3–6 September 2017, Prague, Czech Republic*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds. IEEE Xplore Digital Library, 2017, pp. 1077–1084. [Online]. Available: http://ieeexplore-1ieee-1org-1000047tb1e9c.wbg2.bg.agh.edu.pl/document/8104687/

[B19] R. Klimek, Z. Drwiła, and P. Dzienisik, "Proposal of a smart gun system supporting police interventions," in *Proceedings of 17th International Conference on Arificial Inteligence and Soft Computing (ICAISC 2018), 3–7 June, 2018, Zakopane, Poland*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, Eds., vol. 10842. Springer International Publishing, 2018, pp. 677–688.

[B20] R. Klimek, "System for building and analyzing preference models based on social networking data and SAT solvers," in *Proceedings of 17th International Conference on Arificial Inteligence and Soft Computing (ICAISC 2018), 3–7 June, 2018, Zakopane, Poland*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz, R. Tadeusiewicz, and J. M. Zurada, Eds., vol. 10842. Springer International Publishing, 2018, pp. 387–397.

[B21] ——, "Mapping population and mobile pervasive datasets into individual behaviours for urban ecosystems," in *Proceedings of 15th International Conference on Arti-*

ficial Intelligence and Soft Computing (ICAISC 2016), 12–16 June, 2016, Zakopane, Poland, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., vol. 9692. Springer Verlag, 2016, pp. 683–694.

[B22] ——, "Visualization of logical formulas," in *Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS 2018), 9–12 September 2018, Poznań, Poland*, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds. IEEE Xplore Digital Library, 2018, pp. 419–424.

[B23] R. Klimek, K. Grobler-Dębska, and E. Kucharska, "System for automatic generation of logical formulas," in *III International Conference of Computational Methods in Engineering Science (CMES 2018). Kazimierz Dolny, Poland, November 22–24, 2018*, ser. MATEC Web of Conferences, vol. 252, 2019.

Prace innych autorów

[C24] D. E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 6: Satisfiability*, 1st ed. Addison-Wesley Professional, 2015.

[C25] C. P. Gomes, H. A. Kautz, A. Sabharwal, and B. Selman, "Satisfiability solvers," in *Handbook of Knowledge Representation*, ser. Foundations of Artificial Intelligence, F. van Harmelen, V. Lifschitz, and B. W. Porter, Eds. Elsevier, 2008, vol. 3, pp. 89–134.

[C26] A. Biere, M. Heule, H. van Maaren, and T. Walsh, *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2009.

[C27] R. Schmidt, "AiML.NET. Advances in modal logic. Accessible theorem provers, webpage. `http://www.cs.man.ac.uk/~schmidt/tools/`," 2019, accessed on 11-Mar-2019.

[C28] D. Le Berre, "SAT Live! keep up to date with research on the satisfiability problem. Solvers, web page. `http://www.satlive.org/solvers/`," 2019, accessed on 11-Mar-2019.

[C29] M. Heule, M. Järvisalo, and M. Suda, "The international SAT competitions, web page. `http://www.satcompetition.org/`," 2019, accessed on 11-Mar-2019.

[C30] N. Manthey, "Towards next generation sequential and parallel SAT solvers," *KI - Künstliche Intelligenz*, vol. 30, no. 3, pp. 339–342, Oct 2016.

[C31] M. Broy, "On the role of logic and algebra in software engineering," in *Mathematics, Computer Science and Logic – A Never Ending Story: The Bruno Buchberger Festschrift*, P. Paule, Ed. Springer International Publishing, 2013, pp. 51–68.

[C32] N. Shankar, "Automated deduction for verification," *ACM Computing Surveys*, vol. 41, no. 4, pp. 20:1–20:56, 2009.

[C33] C. Dubois, P. Masci, and D. Méry, Eds., *Proceedings of the 3rd Workshop on Formal Integrated Development Environment, Limassol, Cyprus, November 8, 2016*, ser. Electronic Proceedings in Theoretical Computer Science, vol. 240. Open Publishing Association, 2016. [Online]. Available: http://eptcs.web.cse.unsw.edu.au/content.cgi?FIDE2016

[C34] T. Pender, *UML Bible*. John Wiley & Sons, 2003.

[C35] T. Sunaga, "Theory of an interval algebra and its application to numerical analysis [reprint of res. assoc. appl. geom. mem. 2 (1958), 29–46]," *Japan Journal of Industrial and Applied Mathematics*, vol. 26, no. 2–3, pp. 125–143, 2009.

[C36] S. Tripakis, "Compositionality in the science of system design," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 960–972, 2016.

[C37] ——, "Compositional model-based system design and other foundations for mastering change," in *Transactions on Foundations for Mastering Change I*, ser. Lecture Notes in Computer Science, B. Steffen, Ed., vol. 9960. Springer, 2016, pp. 113–129.

[C38] F. J. Pelletier, "The principle of semantic compositionality," *Topoi*, vol. 13, no. 1, pp. 11–24, 1994.

[C39] Z. G. Szabó, *Problems of Compositionality*. Routledge, 2014.

[C40] A. van Lamsweerde, *Requirements Engineering - From System Goals to UML Models to Software Specifications*. John Wiley & Sons, 2009.

[C41] K. Pohl, *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer Publishing Company, 2010.

[C42] J. C. Augusto, V. Callaghan, D. Cook, A. Kameas, and I. Satoh, "Intelligent Environments: a manifesto," *Human-centric Computing and Information Sciences*, vol. 3, no. 1, p. 12, Jun 2013.

[C43] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Computing Survey*, vol. 43, no. 3, pp. 16:1–16:43, apr 2011.

[C44] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in *Workshop on The What, Who, Where, When, and How of Context-Awareness (CHI 2000)*, April 2000. [Online]. Available: http://www.cc.gatech.edu/fce/contexttoolkit/

[C45] J. Augusto, A. Aztiria, D. Kramer, and U. Alegre, "A survey on the evolution of the notion of context-awareness," *Applied Artificial Intelligence*, vol. 31, no. 7-8, pp. 613–642, 2017.

Rodolfo Klimek