# Rozdział 2

## Problem spełnialności i syntezy przy modelowaniu strategii afektywnych w systemach wieloagentowych

Magdalena Kacprzak*

**Streszczenie.** Praca proponuje nowatorskie podejście do problemów spełnialności i syntezy, rozszerzając ich klasyczne zastosowania na obszar modelowania oraz automatycznej konstrukcji afektywnych systemów wieloagentowych. Celem jest wykazanie, w jaki sposób metoda testowania spełnialności oraz syntezy formuł w logice ATL (ang. *Alternating-time Temporal Logic*) może zostać zaadaptowana do projektowania systemów, w których agenci nie tylko posiadają własne strategie działania, lecz również potrafią rejestrować, przechowywać i dynamicznie modyfikować swoje stany emocjonalne. Zastosowana metoda jest zaimplementowana w narzędziu MsATL i wykorzystuje techniki SMMT (ang. *SAT Modulo Monotonic Theories*) stosowane do weryfikacji boolowskich teorii monotonicznych.

Główne wyzwanie badawcze polega na opracowaniu metodologii umożliwiającej automatyczną konstrukcję modeli spełniających specyfikacje formalne interpretowane w strukturach pozwalających na rozumowanie o strategiach afektywnych. Podejście to integruje aspekty emocjonalne z analizą zachowań strategicznych, co otwiera nowe możliwości w projektowaniu systemów kognitywnych.

Zaproponowana metoda może stanowić fundament narzędzia do automatycznej syntezy afektywnych systemów wieloagentowych, ze szczególnym uwzględnieniem zastosowań w obszarach sztucznej inteligencji emocjonalnej (ang. *affective computing*), agentów społecznych oraz systemów wspierających interakcje człowiek-maszyna. Potencjał praktyczny rozwiązania zilustrowano na przykładzie systemu wspomagania kierowcy w inteligentnych pojazdach wyposażonych w moduły rozpoznawania i interpretacji stanów emocjonalnych.

**Słowa kluczowe:** spełnialność, synteza, strategie afektywne, logika ATL

* Wydział Informatyki, Politechnika Białostocka, Wiejska 45A, 15-351 Białystok, *m.kacprzak@pb.edu.pl*
DOI 10.24427/978-83-68673-08-1_2

## Wprowadzenie

Zagadnienia spełnialności i syntezy mają długą historię, sięgającą początków logiki matematycznej oraz teorii obliczeń. Choć intuicje stojące za nimi są stosunkowo proste – odpowiednio: „Czy coś jest możliwe?" oraz „Czy coś da się zbudować?" – to ich formalizacja i zastosowania okazały się fundamentalne dla rozwoju wielu dziedzin informatyki teoretycznej.

Problem spełnialności formuł logicznych po raz pierwszy pojawił się w kontekście logiki zdań i logiki pierwszego rzędu. Już na początku XX wieku zagadnienie to było przedmiotem badań Fregego, Hilberta, a później Gödla. Formalnie problem spełnialności został osadzony w teorii modeli i teorii dowodów, gdzie spełnialność (ang. *satisfiability*) oznaczała istnienie modelu, w którym dana formuła jest prawdziwa Enderton (2001).

W 1971 roku Stephen Cook sformułował problem spełnialności dla logiki zdaniowej (tzw. SAT) jako pierwszy problem *NP-zupełny*, dając tym samym początek teorii złożoności obliczeniowej i tzw. klasy NP Cook (1971). Od tego momentu problem SAT stał się punktem odniesienia dla oceny trudności innych problemów i impulsem dla rozwoju algorytmiki. Równolegle rozwijały się techniki automatycznego rozumowania oparte m.in. na przeszukiwaniu, rezolucji czy tablicach semantycznych.

Od lat 90. XX wieku obserwujemy gwałtowny rozwój tzw. *SAT-solverów* – wysoko zoptymalizowanych narzędzi do sprawdzania spełnialności dużych instancji formuł zdaniowych. Nowoczesne algorytmy, takie jak DPLL (ang. *Davis-Putnam-Logemann-Loveland*) czy CDCL (ang. *Conflict-Driven Clause Learning*), pozwalają rozwiązywać praktyczne problemy weryfikacji, planowania, testowania i optymalizacji w skali przemysłowej Biere, Heule, van Maaren i Walsh (2021); Biere, Järvisalo i Kiesl (2021). SAT stał się także podstawą dla silników SMT (ang. *Satisfiability Modulo Theories*), które rozszerzają klasyczny SAT o arytmetykę, teorię zbiorów, bitów i inne dziedziny matematyczne.

Z kolei problem syntezy ma swoje korzenie w teorii automatów i programowaniu logicznym. Intuicyjnie odpowiada na pytanie: „Czy da się automatycznie wygenerować system, który działa zgodnie z daną specyfikacją?". W latach 60. i 70. prace Churcha i Büchiego doprowadziły do powstania formalnych metod opisu i konstruowania automatów stanów na podstawie zadanych warunków logicznych.

Kamieniem milowym było sformułowanie przez Amira Pnueliego i Roniego Rosnera w 1989 roku problemu syntezy dla logiki LTL (ang. *Linear Temporal Logic*) Pnueli i Rosner (1989). Autorzy zaproponowali konstrukcję systemów reaktywnych, których zachowanie spełnia zadane wymagania temporalne, na przykład: „Jeśli pojawi się żądanie, to w końcu nastąpi odpowiedź.". W ten sposób rozpoczęto badania nad automatyczną konstrukcją kontrolerów i strategii działania.

Od tego czasu problem syntezy został rozwinięty w wielu kierunkach. Powstały logiki strategiczne, na przykład ATL (ang. *Alternating-time Temporal Logic*) czy SL (ang. *Strategy Logic*), które pozwalają opisywać możliwości działania agentów

w warunkach współzależności i rywalizacji Alur, Henzinger i Kupferman (2002); Mogavero, Murano, Perelli i Vardi (2014a). Wprowadzono również pojęcie syntezy ograniczonej (ang. *bounded synthesis*), czyli syntezy przy ograniczeniach na wielkość lub strukturę systemu Finkbeiner i Schewe (2013). W systemach cyberfizycznych i robotyce rozwijane są metody syntezy sterowania z uwzględnieniem dynamiki ciągłej, czasu rzeczywistego oraz ograniczeń fizycznych Tabuada (2009).

Choć problemy spełnialności i syntezy są odmienne w formalizacji, mają wspólne podłoże filozoficzne. Oba dotyczą realizowalności logicznych warunków: spełnialność pyta o możliwość, synteza zaś o wykonanie. Z technicznego punktu widzenia synteza jest problemem konstrukcyjnym – wymaga wygenerowania obiektu (strategii, automatu, programu), który spełnia dane wymagania w sposób niezawodny, często w kontekście interakcji z otoczeniem. Spełnialność zaś sprowadza się do stwierdzenia istnienia rozwiązania, bez wymogu jego konstrukcji.

Obecnie oba zagadnienia są podstawą współczesnej weryfikacji formalnej, logiki komputerowej, teorii gier i planowania automatycznego. Łączą idee z zakresu matematyki, informatyki, inżynierii systemów oraz sztucznej inteligencji i stanowią aktywne obszary badań zarówno teoretycznych, jak i stosowanych.

### Zastosowania praktyczne problemów spełnialności i syntezy

Problemy spełnialności i syntezy mają nie tylko fundamentalne znaczenie teoretyczne, ale także liczne zastosowania w rzeczywistej inżynierii systemów, analizie programów, sztucznej inteligencji i automatyce Biere, Heule i in. (2021); E. M. Clarke, Grumberg, Kroening, Peled i Veith (2018).

W sztucznej inteligencji spełnialność znajduje zastosowanie w zadaniach planowania, gdzie scenariusze działania reprezentuje się jako zestaw reguł logicznych, a cel formułuje jako końcową formułę logiczną. Znalezienie planu odpowiada wówczas znalezieniu spełnialnego przypisania zmiennych, które realizuje cel, zachowując reguły przejść. Takie podejścia stosuje się na przykład w robotyce (planowanie ruchów robota) czy logistyce (optymalizacja ścieżek dostaw) Ghallab, Nau i Traverso (2004).

Innym obszarem wykorzystania spełnialności są systemy detekcji błędów i eksploracji stanów, na przykład. w testowaniu bezpieczeństwa, gdzie sprawdza się, czy możliwa jest nieautoryzowana sekwencja zdarzeń, która łamie wymagania systemowe. W tym kontekście formuły opisujące niepożądane stany są przekształcane do problemu SAT lub SMT, co umożliwia automatyczne znalezienie luk i błędów w systemach Biere, Heule i in. (2021).

Problem syntezy dotyczy natomiast sytuacji, w których nie tylko sprawdzamy, czy coś jest możliwe, ale aktywnie konstruujemy system (np. automat, program, strategię) spełniający daną specyfikację logiczną. Klasycznym przykładem jest *synteza reaktywnych systemów sterowania*, w której na podstawie formuły w logice LTL generowany jest automat reagujący w sposób ciągły na dane wejściowe, realizując

wymagania bezpieczeństwa i żywotności Pnueli i Rosner (1989). Taki automat może następnie zostać przekształcony w kod sterownika przemysłowego lub sprzęt cyfrowy (np. kontroler FPGA) Bloem, Jobstmann, Piterman, Pnueli i Saar (2012).

Syntezę stosuje się również w projektowaniu protokołów komunikacyjnych. Zamiast ręcznie implementować system można zdefiniować jego wymagania (np. niezawodność, brak zakleszczeń, synchronizacja) w logice temporalnej, a następnie automatycznie wygenerować implementację, która spełnia te wymagania. Takie podejścia są coraz częściej rozważane w projektowaniu systemów krytycznych, na przykład lotniczych lub medycznych, gdyż ręczne projektowanie jest podatne na błędy E. M. Clarke i in. (2018).

W ostatnich latach duże zainteresowanie budzi synteza strategii w systemach wieloagentowych i grach. W tym kontekście formuły logiczne, na przykład w ATL lub SL, wyrażają cele, które agenci mają osiągać niezależnie od zachowań innych graczy. Problem polega na skonstruowaniu strategii (lub zespołu strategii) dla jednego lub kilku agentów, które zapewniają osiągnięcie celu w obliczu różnych zachowań przeciwnika. Zastosowania obejmują m.in. sztuczną inteligencję w grach komputerowych, negocjacje międzyagentowe, kontrolę autonomicznych pojazdów oraz bezpieczeństwo systemów rozproszonych Alur i in. (2002); Mogavero, Murano, Perelli i Vardi (2014b).

### Wkład pracy i główne rezultaty

Niniejsza praca wnosi nową perspektywę do zastosowań problemów spełnialności i syntezy, rozszerzając ich wykorzystanie na obszar modelowania i automatycznej konstrukcji afektywnych systemów wieloagentowych. Celem jest opracowanie formalnego podejścia umożliwiającego projektowanie i analizę systemów, w których agenci nie tylko posiadają własne strategie działania, ale także potrafią rejestrować, przechowywać oraz dynamicznie modyfikować swoje stany emocjonalne.

Podstawowym wyzwaniem badawczym jest zaproponowanie metody formalizacji takich systemów oraz pokazanie, w jaki sposób można automatycznie konstruować modele odpowiadające specyfikacji zapisanej w logice formalnej. Punktem wyjścia jest przede wszystkim opis systemu w języku ATL Alur i in. (2002), a celem wygenerowanie modelu wieloagentowego spełniającego zadane własności związane z afektywnymi strategiami.

Zastosowanie problemu spełnialności umożliwia weryfikację poprawności oraz niesprzeczności specyfikacji. Dzięki temu można zidentyfikować potencjalne konflikty logiczne przed przystąpieniem do konstrukcji systemu. Z kolei synteza pozwala na automatyczne wygenerowanie modelu systemu z uwzględnieniem parametrów semantycznych wybranych na potrzeby konkretnego zastosowania Mogavero i in. (2014b); Pnueli i Rosner (1989).

Jednym z kluczowych aspektów omawianej metody jest elastyczność w zakresie semantyki: system może być synchroniczny lub asynchroniczny, agenci mogą mieć wiedzę globalną lub tylko lokalną, a decyzje mogą być podejmowane zarówno na podstawie bieżącego stanu, jak i historii przebiegu działania systemu. Dzięki temu możliwe jest modelowanie szerokiego wachlarza scenariuszy komunikacji i współpracy między agentami. W przypadku ręcznej konstrukcji modeli zmiana semantyki wymagałaby istotnych i czasochłonnych modyfikacji. Tymczasem automatyczne podejście pozwala na szybką i wygodną zmianę semantycznej interpretacji bez konieczności rekonstrukcji całego systemu.

W rezultacie opracowana metoda może stanowić podstawę narzędzia do automatycznej syntezy modeli afektywnych systemów wieloagentowych. Warto podkreślić, że celem nie jest generowanie strategii emocjonalnych *sensu stricto*, lecz budowa modelu systemu, w którym agenci działają zgodnie z określonymi afektywnymi strategiami. Model ten powinien spełniać z góry określone własności, na przykład dotyczące zależności między stanami emocjonalnymi a możliwymi działaniami.

Podejście to może znaleźć zastosowanie w badaniach nad sztuczną inteligencją emocjonalną (ang. *affective computing*) Picard (1997), projektowaniu agentów społecznych i emocjonalnych, a także w symulacjach interakcji człowiek-maszyna, gdzie emocje odgrywają istotną rolę w podejmowaniu decyzji i współpracy.

## 2.1. Strategie afektywne

Strategia afektywna to świadome działanie ukierunkowane na regulację emocji, motywacji i nastawienia, które wpływają na przebieg i skuteczność różnych działań. Jest to sposób zarządzania emocjami mający na celu zwiększenie motywacji, zmniejszenie stresu, wzmacnianie pozytywnego nastawienia, podtrzymywanie zaangażowania oraz budowanie wiary w siebie. W ten sposób strategie afektywne mogą wspierać procesy uczenia się, redukować negatywne emocje (takie jak lęk czy frustracja), a w konsekwencji zwiększyć efektywność różnych działań poprzez kontrolę nad afektem.

Symulowanie emocji stanowi istotny i często integralny element wielu współczesnych systemów sztucznej inteligencji. Jednym z głównych celów AI w obszarze interakcji człowiek-komputer jest jak najwierniejsze odwzorowanie ludzkich zachowań, w których emocje odgrywają kluczową rolę. Systemy naśladujące człowieka muszą być zdolne do generowania reakcji emocjonalnych zgodnych z kontekstem sytuacyjnym, a równocześnie powinny rozpoznawać emocje użytkownika i adekwatnie na nie reagować, na przykład poprzez dostosowanie tonu wypowiedzi, treści komunikatu czy sposobu działania. Wymaga to nie tylko technicznej implementacji odpowiednich mechanizmów, ale także formalizacji pojęć związanych z afektem, takich jak emocje, nastroje, intencje czy empatia. To właśnie ta potrzeba wzajemnego

zrozumienia i spójnego zachowania na linii człowiek-maszyna sprawia, że temat modelowania emocji zyskał tak duże znaczenie w środowisku badawczym zajmującym się sztuczną inteligencją, robotyką społeczną czy systemami adaptacyjnymi Brave i Nass (2002); Picard (1997); Tao i Tan (2005).

### 2.1.1. Strategie afektywne w kontekście wspomagania kierowcy

W inteligentnych pojazdach (ang. *smart cars*) coraz większą rolę odgrywają strategie afektywne ukierunkowane na poprawę komfortu i bezpieczeństwa kierowcy Healey i Picard (2005); Zepf, Hernandez, Schmitt, Minker i Picard (2020). Ich celem jest rozpoznanie aktualnego stanu emocjonalnego kierowcy i jego modyfikacja przy użyciu odpowiednich bodźców, np. muzyki, światła czy komunikatów. Przykładem mogą być:

- odtworzenie relaksującej muzyki w sytuacjach stresujących (np. w korku drogowym),
- włączenie dynamicznej playlisty w celu zwiększenia czujności w przypadku wykrycia senności,
- przypomnienie o konieczności odpoczynku w przypadku długotrwałego napięcia.

Takie działania mają na celu nie tylko poprawę dobrostanu kierowcy, ale również zwiększenie bezpieczeństwa jazdy poprzez wpływ na jego zachowanie w czasie rzeczywistym. Wspierając emocje kierowcy, pojazd staje się aktywnym uczestnikiem procesu jazdy, reagującym nie tylko na warunki drogowe, ale również na *afektywny stan użytkownika*. Implementacja takich strategii wymaga integracji modułów rozpoznawania emocji (na podstawie mimiki, głosu, stylu jazdy) z systemami interaktywnymi pojazdu. W efekcie powstaje spersonalizowane środowisko jazdy dopasowane do indywidualnych potrzeb emocjonalnych użytkownika.

### 2.1.2. Integracja strategii afektywnych z systemem rozpoznawania emocji Affectiva Automotive AI

*Affectiva Automotive* AI to zaawansowany system sztucznej inteligencji opracowany przez firmę Affectiva (obecnie część Smart Eye), który umożliwia monitorowanie emocji oraz poziomu uwagi kierowcy w czasie rzeczywistym. System wykorzystuje kamerę skierowaną na twarz użytkownika oraz algorytmy uczenia maszynowego do analizy mikromimiki, kierunku spojrzenia, pozycji głowy oraz oznak zmęczenia i rozproszenia Affectiva2025 (2025); Eye (2025).

Dzięki tej analizie możliwe jest wykrywanie emocji takich jak złość, smutek, zaskoczenie czy radość, a także ocena zaangażowania kierowcy. Affectiva AI różni się

od klasycznych systemów bezpieczeństwa, ponieważ dostarcza danych poznawczych i emocjonalnych, które mogą być wykorzystywane przez inne komponenty pojazdu, na przykład systemy rekomendacyjne. Integracja Affectiva AI z modułem afektywnych strategii pozwala dynamicznie dostosowywać środowisko jazdy: oświetlenie, interfejs, dźwięki czy rekomendacje przerw, w zależności od wykrytego psychofizycznego stanu kierowcy.

Technologia Affectiva AI znajduje zastosowanie zarówno w pojazdach autonomicznych, jak i konwencjonalnych, stanowiąc ważny element systemów monitorowania kierowcy (ang. *Driver Monitoring Systems*). Jej skuteczność została potwierdzona w badaniach z udziałem producentów samochodów w rzeczywistych warunkach drogowych. Stanowi przykład zastosowania obliczeń afektywnych (ang. *affective computing*) w kontekście poprawy komfortu i bezpieczeństwa jazdy.

## 2.2. Systemy wieloagentowe

System wieloagentowy (ang. *mulit-agent system*, MAS) to system złożony z wielu współdziałających agentów, którymi mogą być zarówno byty programowe, jak i roboty. Agenci ci są autonomiczni, co oznacza, że potrafią samodzielnie podejmować decyzje w oparciu o własne cele i posiadaną wiedzę. Każdy agent posiada określone atrybuty, takie jak lokalne stany i działania. Zasady jego zachowania, reprezentowane przez lokalny protokół i lokalną funkcję przejścia, określają, jak agent działa w zależności od swojego lokalnego stanu (i środowiska).

Systemy wieloagentowe oferują potężny paradygmat rozwiązywania złożonych, rozproszonych problemów. Poprzez zdefiniowanie agentów, ich interakcji oraz środowiska możemy tworzyć modele, które symulują i rozwiązują rzeczywiste problemy w różnych dziedzinach. Sukces takich systemów zależy od skutecznej koordynacji, komunikacji i strategii decyzyjnych dostosowanych do konkretnej aplikacji.

**Definicja 2.1 (MAS).** *System wieloagentowy* $\mathscr{MAS}_n$ składa się z $n$ agentów $\mathscr{A}gt = \{1,\ldots,n\}$*, gdzie każdemu agentowi $i \in \mathscr{A}gt$ odpowiada 7-krotka $AG_i = (Loc_i, \iota_i, Act_i, P_i, T_i, AP_i, \mathscr{V}_i)$ zawierająca:

- skończony, niepusty zbiór *lokalnych stanów* $Loc_i = \{\ell_i^1, \ell_i^2, \ldots, \ell_i^{n_i}\}$;
- *początkowy lokalny stan* $\iota_i \in Loc_i$;
- skończony, niepusty zbiór *lokalnych akcji* $Act_i = \{a_i^1, a_i^2, \ldots, a_i^{m_i}\}$;
- *lokalny protokół* $P_i : Loc_i \to 2^{Act_i} \setminus \{\emptyset\}$ definujący dostępne akcje w każdym stanie - zakładamy, że zbiór akcji dostępnych w dowolnym stanie nie może być pusty;
- (częściową) *lokalną funkcję przejścia* $T_i : Loc_i \times Act \to Loc_i$ taką, że jeśli $T_i(\ell_i, \alpha)$ jest określona, to $\alpha^i \in P_i(\ell_i)$, gdzie $Act \triangleq \prod_{i \in \mathscr{A}gt} Act_i$ to zbiór *globalnych ak-*

* Komponent środowiska może być tutaj dodany bez zwiększenia trudności technicznych.

*cji* wszystkich agentów, natomiast $\alpha^i$ to akcja lokalna agenta $i \in \mathscr{A}gt$ będąca składową globalnej akcji $\alpha \in Act$;
- skończony, niepusty zbiór *lokalnych zmiennych atomowych* $AP_i = \{\mathsf{p}_i^1, \mathsf{p}_i^2, \ldots, \mathsf{p}_i^{r_i}\}$;
- *lokalną funkcję wartościującą* $\mathscr{V}_i : Loc_i \to 2^{AP_i}$.

Zauważmy, że rozważamy *systemy wieloagentowe synchroniczne*, w których każda *globalna akcja* to $n$-krotka $\langle a_i \rangle_{i \in \mathscr{A}gt}$, gdzie $a_i \in Act_i$, czyli każdy agent wykonuje jedną lokalną akcję.

### 2.2.1. Model

Model systemu wieloagentowego dostarcza spójnych i formalnych ram do opisu zachowań agentów, umożliwiając precyzyjną analizę funkcjonowania całego systemu, szczególnie w kontekstach wymagających koordynacji. Skupia się na lokalnych działaniach i interakcjach pomiędzy agentami, co pozwala uchwycić złożoną dynamikę ich współdziałania. W niniejszym podejściu przyjmujemy model oparty na strukturze systemu interpretowanego (ang. *Interpreted System*, IS), który stanowi naturalne środowisko dla analizy wiedzy oraz zmienności stanów lokalnych.

**Definicja 2.2 (Model).** Niech $\mathscr{MAS}_n$ będzie systemem wieloagentowym. *Model* tego systemu to 4-krotka $\mathscr{M} = (\mathscr{Q}, \iota, T, \mathscr{V})$, gdzie:

- $\mathscr{Q} = Loc_1 \times \cdots \times Loc_n$ to zbiór globalnych *stanów*;
- $\iota = (\iota_1, \ldots, \iota_n) \in \mathscr{Q}$ to *początkowy stan globalny*;
- $T : \mathscr{Q} \times Act \to \mathscr{Q}$ to (częściowa) *globalna funkcja przejścia*, taka że $T(q, \alpha) = q'$ wtedy i tylko wtedy, gdy $T_i(q_i, \alpha) = q'_i$ dla każdego $i \in \mathscr{A}gt$, gdzie dla globalnego stanu $q = (\ell_1, \ldots, \ell_n)$, przy czym przez $q_i = \ell_i$ oznaczamy lokalną składową agenta $i$;
- $\mathscr{V} : \mathscr{Q} \to 2^{AP}$ to funkcja wartościowania, taka że $AP \triangleq \bigcup_{i=1}^n AP_i$ jest sumą lokalnych zmiennych atomowych oraz $\mathscr{V}((\ell_1, \ldots, \ell_n)) = \bigcup_{i=1}^n \mathscr{V}_i(\ell_i)$.

Mówimy, że akcja $\alpha \in Act$ jest *dozwolona* w stanie $q \in \mathscr{Q}$ wtedy, gdy $T(q, \alpha) = q'$ dla pewnego $q' \in \mathscr{Q}$. Zakładamy, że w każdym stanie $q \in \mathscr{Q}$ istnieje przynajmniej jedna dozwolona akcja, tzn. dla każdego $q \in \mathscr{Q}$ istnieją $\alpha \in Act$ i $q' \in \mathscr{Q}$, takie że $T(q, \alpha) = q'$.

*Ścieżka* to (skończony lub nie) ciąg globalnych stanów $\pi = q_0, q_1, \ldots \in \mathscr{Q}^* \cup \mathscr{Q}^\omega$, taki że dla każdego $j \geq 0$ zachodzi $\pi[j+1] = T(\pi[j], \alpha_j)$ dla pewnej globalnej akcji $\alpha_j \in Act$, gdzie $\pi[j]$ oznacza $j$-ty stan w ścieżce $\pi$. Definiujemy też *przebieg* (ang. *run*) jako naprzemienny (skończony lub nieskończony) ciąg globalnych stanów i akcji $(q_0, \alpha_0, q_1, \alpha_1, q_2, \alpha_2, \ldots)$, taki że dla każdego $j \geq 0$ zachodzi $q_{j+1} = T(q_j, \alpha_j)$. Przebieg odzwierciedla możliwą ewolucję systemu w czasie. Każdemu przebiegowi

odpowiada dokładnie jedna ścieżka, którą uzyskujemy przez projekcję przebiegu na jego komponenty stanowe. Zbiór wszystkich przebiegów w modelu $\mathscr{M}$ zaczynających się w stanie $q$ oznaczamy $\Pi_{\mathscr{M}}(q)$.

### 2.2.2. Typy systemów wieloagentowych

Systemy wieloagentowe można modelować na różne sposoby w zależności od przyjętych założeń dotyczących synchronizacji działań, struktury stanów, dostępnej wiedzy oraz sposobu reprezentacji akcji. Kluczowe różnice występują między systemami synchronicznymi i asynchronicznymi oraz strukturami typu CGS (ang. *Concurrent Game Structures*) i systemami interpretowanymi IS (ang. *Interpreted Systems*).

W modelach synchronicznych agenci wykonują akcje równocześnie, co upraszcza analizę i sprzyja stosowaniu logik takich jak ATL. Modele asynchroniczne pozwalają na realistyczne odwzorowanie systemów rozproszonych, ale są trudniejsze w analizie. Struktury CGS opierają się na lokalnych akcjach agentów i globalnych stanach, umożliwiając precyzyjne definiowanie strategii. W systemach interpretowanych IS podstawą są lokalne stany agentów i semantyka wiedzy, co sprzyja modelowaniu epistemiki i ograniczonej obserwowalności. Zmienne atomowe mogą być globalne (wspólne) lub lokalne (prywatne dla agenta), co wpływa na możliwości poznawcze oraz sposób formułowania i interpretacji specyfikacji.

Wybór konkretnego modelu – CGS vs. IS, synchroniczny vs. asynchroniczny, lokalny vs. globalny – zależy od celu analizy: projektowania strategii, modelowania wiedzy czy odwzorowania realistycznych warunków działania systemu.

Proponowane w pracy podejście zostało zaimplementowane w kontekście systemów synchronicznych, w których każdemu agentowi przypisane są lokalnie: stany, akcje oraz zmienne atomowe, a wspólne działanie agentów modelowane jest za pomocą struktury systemu interpretowanego. Choć rozwiązanie to zostało przedstawione na przykładzie konkretnej klasy systemów wieloagentowych, może być z powodzeniem zastosowane do dowolnego typu MAS. Wymaga to jedynie niewielkich modyfikacji w sposobie kodowania modelu, lecz nie wiąże się z dodatkowymi trudnościami technicznymi. Sama procedura syntezy przebiega w sposób analogiczny.

## 2.3. Strategiczna logika temporalna

Logika czasu dyskretnego dzieli się na trzy główne typy: liniową logikę czasową, która zakłada domyślną kwantyfikację uniwersalną nad wszystkimi ścieżkami generowanymi przez wykonanie systemu (ang. *Linear Temporal Logic*, LTL) Baier i Katoen (2008); Pnueli (1977), logikę na rozgałęzionej strukturze czasu, która pozwala na

jawne kwantyfikowanie egzystencjalne i uniwersalne nad wszystkimi ścieżkami (ang. *Computation Tree Logic*, CTL) E. Clarke i Emerson (1981), oraz bardziej ogólną alternującą logikę czasową (ang. *Alternating-time Temporal Logic*, ATL) Alur, Henzinger i Kupferman (1997). Ta ostatnia umożliwia selektywną kwantyfikację nad ścieżkami wynikającymi z gier, w których system i środowisko na przemian wykonują ruchy. Podczas gdy logiki liniowa i rozgałęziająca są odpowiednie do specyfikacji systemów zamkniętych, logika ATL idealnie nadaje się do opisu systemów otwartych. Logikę tę zastosujemy też do specyfikowania wymagań afektywnych systemów wieloagentowych.

### 2.3.1. Składnia

Logika ATL została zaproponowana przez Rajeeva Alura, Thomasa A. Henzingera i Orna Kupfera w 1997 roku Alur i in. (1997, 2002) jako rozszerzenie logik czasowych umożliwiające wyrażanie własności systemów wieloagentowych. W przeciwieństwie do LTL czy CTL logika ATL pozwala mówić o możliwościach grup agentów do wymuszania spełnienia formuł poprzez swoje strategie. ATL jest więc logicznym formalizmem dla rozumowania o grach, strategiach i możliwościach koalicji agentów.

W podstawowej wersji logiki ATL (ang. *vanilla ATL*) każde wystąpienie modalności strategicznej jest bezpośrednio poprzedzone przez operator temporalny.

**Definicja 2.3 (Składnia logiki ATL).**
Niech $AP$ będzie skończonym zbiorem zmiennych atomowych, a $\mathscr{A}gt$ – skończonym zbiorem agentów. Język $\mathscr{L}$ logiki ATL definiuje się za pomocą następującej gramatyki:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle\Gamma\rangle\rangle X\varphi \mid \langle\langle\Gamma\rangle\rangle \varphi U \varphi \mid \langle\langle\Gamma\rangle\rangle G\varphi,$$

gdzie $\mathsf{p} \in AP$ oraz $\Gamma \subseteq \mathscr{A}gt$.

Nieformalnie formuła $\langle\langle\Gamma\rangle\rangle\gamma$ wyraża, że grupa agentów $\Gamma$ posiada wspólną strategię pozwalającą wymusić spełnienie własności temporalnej $\gamma$. W formułach wykorzystuje się standardowe operatory temporalne: „$X$” (czyli „w następnym kroku”), „$G$” („zawsze od teraz”) oraz „$U$” („dopóki nie” w wersji silnej). Często stosowanym operatorem jest też $F$ definiowany w następujący sposób: $F\varphi = true\, U\varphi$, gdzie $true$ jest stałą logiczną, która jest spełniona w każdym modelu i każdym stanie. Operator ten oznacza, że formuła będzie prawdziwa w pewnym momencie w przyszłości. Jest to tzw. operator „w końcu”.

### 2.3.2. Strategie

Niech $\mathscr{M}$ będzie modelem. *Strategia* agenta $i \in \mathscr{A}gt$ w $\mathscr{M}$ to plan określający, co agent $i$ powinien zrobić w każdej możliwej sytuacji. Możliwe są różne warianty semantyczne strategii. W niniejszej pracy koncentrujemy się na strategiach niezależnych od historii, czyli bezpamięciowych (ang. *memoryless*) przy pełnej oraz niepełnej informacji. Formalnie:

- *Strategia bezpamięciowa przy pełnej informacji* dla agenta $i$ to funkcja $\sigma_i \colon \mathscr{Q} \to Act_i$, taka że $\sigma_i(q) \in P_i(q^i)$ dla każdego $q \in \mathscr{Q}$. Zbiór takich strategii oznaczamy przez $\Sigma_{\mathrm{Ir}}$.
- *Strategia bezpamięciowa przy niepełnej informacji* dla agenta $i$ to funkcja $\sigma_i \colon Loc_i \to Act_i$, taka że $\sigma_i(l) \in P_i(l)$ dla każdego $\ell \in Loc_i$. Zbiór takich strategii oznaczamy przez $\Sigma_{\mathrm{ir}}$.

Zatem strategia przy pełnej informacji może przypisywać różne akcje różnym stanom globalnym, natomiast przy niepełnej informacji decyzje agenta zależą wyłącznie od jego lokalnego stanu. Za Schobbens (2004) nazywamy te strategie odpowiednio *Ir-strategiami* oraz *ir-strategiami*. Oprócz strategii bezpamięciowych wyróżnia się też strategie zależne od historii, w których decyzja agenta dotycząca wyboru akcji opiera się nie tylko na bieżącym stanie, lecz również na wcześniejszym przebiegu systemu. Taka historia może obejmować ciąg stanów globalnych lub lokalnych, w zależności od zakresu wiedzy dostępnej danemu agentowi.

*Strategia zespołowa (koalicyjna)* $\sigma_\Gamma$ dla koalicji $\Gamma \subseteq \mathscr{A}gt$ to krotka strategii, po jednej dla każdego agenta $i \in \Gamma$. Zbiory strategii zespołowych przy pełnej i niepełnej informacji oznaczamy odpowiednio przez $\Sigma_\Gamma^{\mathrm{Ir}}$ oraz $\Sigma_\Gamma^{\mathrm{ir}}$. Ponadto, jeśli $\sigma_\Gamma = (\sigma_1, \ldots, \sigma_k)$ jest strategią koalicji $\Gamma = \{i_1, \ldots, i_k\}$, to dla każdego stanu $q \in \mathscr{Q}$ definiujemy następująco $\sigma_\Gamma(q) := (\sigma_1(q), \ldots, \sigma_k(q))$.

Intuicyjnie, wynikiem działania strategii zespołowej $\sigma_\Gamma$, czyli jej realizacji (ang. *outcome*), w stanie globalnym $q$ jest zbiór wszystkich nieskończonych przebiegów, które mogą powstać, gdy w każdym kroku agenci z koalicji $\Gamma$ wykonują akcje zgodne z $\sigma_\Gamma$, a pozostali agenci ($\mathscr{A} \setminus \Gamma$) działają zgodnie ze swoimi dozwolonymi protokołami.

**Definicja 2.4 (Realizacja strategii).** Niech $Y \in \{\mathrm{Ir}, \mathrm{ir}\}$. *Realizacją* strategii $\sigma_\Gamma \in \Sigma_\Gamma^Y$ w stanie $q \in \mathscr{Q}$ nazywamy zbiór $out_{\mathscr{M}}(q, \sigma_\Gamma) \subseteq \Pi_{\mathscr{M}}(q)$, taki że $\pi = q_0, \alpha_0, q_1, \alpha_1, \cdots \in out_{\mathscr{M}}(q, \sigma_\Gamma)$, wtedy i tylko wtedy, gdy $q_0 = q$ oraz dla każdego $i \in \mathbb{N}$ i każdego $j \in \Gamma$ zachodzi: $\alpha_i^j = \sigma_j(\pi[i])$ dla $Y = \mathrm{Ir}$, a także $\alpha_i^j = \sigma_j(\pi[i]^j)$ dla $Y = \mathrm{ir}$.

### 2.3.3. Semantyka

Analiza zachowania systemu względem zadanej specyfikacji logicznej wymaga precyzyjnego określenia, kiedy dana formuła jest prawdziwa w konkretnym stanie modelu. Formalnie semantyka logiki ATL, parametryzowana typem strategii $Y \in \{\mathrm{Ir}, \mathrm{ir}\}$, jest definiowana następująco:

$\mathscr{M}, q \models_Y \mathsf{p}$ wttw $\mathsf{p} \in \mathscr{V}(q)$ dla $\mathsf{p} \in AP$;
$\mathscr{M}, q \models_Y \neg\varphi$ wttw $\mathscr{M}, q \not\models_Y \varphi$;
$\mathscr{M}, q \models_Y \varphi_1 \wedge \varphi_2$ wttw $\mathscr{M}, q \models_Y \varphi_1$ oraz $\mathscr{M}, q \models_Y \varphi_2$;
$\mathscr{M}, q \models_Y \langle\langle\Gamma\rangle\rangle X\varphi$ wttw istnieje strategia $\sigma_\Gamma \in \Sigma_\Gamma^Y$, taka że
$out_{\mathscr{M}}(q, \sigma_\Gamma) \neq \emptyset$ i dla każdej ścieżki $\pi \in out_{\mathscr{M}}(q, \sigma_\Gamma)$ zachodzi
$\mathscr{M}, \pi \models_Y X\varphi$, tzn. $\mathscr{M}, \pi[1] \models_Y \varphi$;
$\mathscr{M}, q \models_Y \langle\langle\Gamma\rangle\rangle \varphi_1 U \varphi_2$ wttw istnieje strategia $\sigma_\Gamma \in \Sigma_\Gamma^Y$, taka że
$out_{\mathscr{M}}(q, \sigma_\Gamma) \neq \emptyset$ i dla każdej ścieżki $\pi \in out_{\mathscr{M}}(q, \sigma_\Gamma)$ zachodzi
$\mathscr{M}, \pi \models_Y \varphi_1 U \varphi_2$, tzn. $\mathscr{M}, \pi[i] \models_Y \varphi_2$ dla pewnego $i \geq 0$ oraz
$\mathscr{M}, \pi[j] \models_Y \varphi_1$ dla każdego $0 \leq j < i$;
$\mathscr{M}, q \models_Y \langle\langle\Gamma\rangle\rangle G\varphi$ wttw istnieje strategia $\sigma_\Gamma \in \Sigma_\Gamma^Y$, taka że
$out_{\mathscr{M}}(q, \sigma_\Gamma) \neq \emptyset$ i dla każdej ścieżki $\pi \in out_{\mathscr{M}}(q, \sigma_\Gamma)$ zachodzi
$\mathscr{M}, \pi \models_Y G\varphi$, tzn. $\mathscr{M}, \pi[i] \models_Y \varphi$, dla każdego $i \geq 0$.

Formalna analiza właściwości opisanych formułami logiki ATL odwołuje się do pojęć prawdziwości i spełnialności. Prawdziwość określa, czy dana formuła jest spełniona w stanie początkowym konkretnego modelu systemu, natomiast spełnialność dotyczy istnienia modelu, w którym formuła jest prawdziwa.

**Definicja 2.5 (Prawdziwość).** Formuła ATL $\varphi$ jest prawdziwa w modelu $\mathscr{M}$, co oznaczamy jako $\mathscr{M} \models_Y \varphi$, jeśli jest spełniona w stanie początkowym tego modelu, tzn. $\mathscr{M}, \iota \models_Y \varphi$ dla $Y \in \{Ir, ir\}$.

**Definicja 2.6 (Spełnialność).** Formuła ATL $\varphi$ jest spełnialna, jeśli istnieje model $\mathscr{M}$, w którym jest ona prawdziwa, tzn. $\mathscr{M} \models_Y \varphi$ dla $Y \in \{Ir, ir\}$.

## 2.4. Reprezentacja emocji w systemach wieloagentowych

Formalne ujęcie emocji w systemach wieloagentowych stanowi fundament do analizy strategii afektywnych, w których emocje wpływają na wybory agentów. Pozwala to modelować bardziej realistyczne zachowania, szczególnie w kontekście interakcji społecznych, symulacji zachowań oraz systemów uczących się.

Modelowanie emocji w kontekście logiki ATL otwiera możliwość wzbogacenia klasycznego rozumowania o strategiach agentów o wymiar afektywny. Analiza strategii afektywnych, czyli takich, które są uwarunkowane stanem emocjonalnym agentów

lub wpływają na ten stan, wymaga formalnego ujęcia emocji. Stany emocjonalne muszą być jednoznacznie reprezentowane zarówno w modelu systemu wieloagentowego, jak i w języku, w którym formułowana jest jego specyfikacja.

### 2.4.1. Modelowanie emocji

W psychologii uznanym zestawem podstawowych emocji są emocje zidentyfikowane przez Paula Ekmana Ekman (1992), które uznaje się za uniwersalne i niezależne od kontekstu kulturowego. Należą do nich: *strach* (ang. *fear*), *wstręt* (ang. *disgust*), *radość* (ang. *joy*), *smutek* (ang. *sadness*) oraz *złość* (ang. *anger*). Są to emocje powszechnie doświadczane przez ludzi i rozpoznawane w podobny sposób na całym świecie. Inne emocje często powstają jako kombinacje tych pięciu podstawowych. W proponowanym podejściu przyjmujemy za Kacprzak (2017), że stan emocjonalny agenta jest reprezentowany przez wektor emocji

$$E = (e_1, e_2, e_3, e_4, e_5),$$

który stanowi uporządkowaną piątkę liczb całkowitych odpowiadających kolejno intensywności emocji typu: **strach**, **wstręt**, **radość**, **smutek** i **złość**. Każdy komponent tego wektora przyjmuje wartości ze zbioru $IE$, który dla uproszczenia może zostać zdefiniowany jako $\{-1, 0, 1\}$, gdzie: $-1$ oznacza niski poziom intensywności danej emocji, 0 – stan neutralny, natomiast 1 – wysoki poziom intensywności.

Oczywiście w bardziej zaawansowanych modelach zbiór $IE$ może zostać rozszerzony do większego zakresu wartości całkowitych (np. od $-k$ do $k$), co pozwala dokładniej uchwycić zróżnicowanie w intensywności emocji. Poziomy emocji zmieniają się w czasie w zależności od działań podejmowanych przez agentów oraz kontekstu sytuacyjnego. Każda akcja może wpływać na intensywność jednej lub więcej emocji w sposób dodatni lub ujemny. Zmiana emocji może być modelowana deterministycznie lub probabilistycznie w zależności od zastosowanej semantyki.

Warto zauważyć, że dopuszcza się jednoczesne występowanie wysokich poziomów różnych emocji, nawet tych pozornie sprzecznych, takich jak radość i złość. Takie „mieszane uczucia" są zgodne z ludzkim doświadczeniem emocjonalnym i mogą odgrywać ważną rolę w strategiach agentów, na przykład motywując do działania mimo frustracji.

### 2.4.2. Afektywny system wieloagentowy i jego model

Wzbogacenie systemu wieloagentowego o aspekt emocji polega na rozszerzeniu stanów lokalnych agentów o komponent wyrażający intensywność podstawowej

piątki emocji. Tym samym wprowadzamy zbiór wektorów emocji, czyli 5-krotek o wartościach z zadanego zbioru. Niech $IE$ formalnie będzie zadanym, ustalonym podzbiorem liczb całkowitych oraz niech

$$SE \triangleq \{(e_1, e_2, e_3, e_4, e_5) \ : \ e_i \in IE \ \text{ dla i} = 1,2,3,4,5\}.$$

Teraz dla agenta definiujemy jego afektywny odpowiednik.

**Definicja 2.7 (Afektywne rozszerzenie).** Niech dla agenta $i \in \mathscr{A}gt$, $AG_i = (Loc_i, \iota_i, Act_i, P_i, T_i, AP_i, \mathscr{V}_i)$. *Afektywne rozszerzenie* zdefiniowane dla tego agenta jest to 7-krotka $EAG_i = (ELoc_i, e\iota_i, Act_i, EP_i, ET_i, EAP_i, \mathscr{E}\mathscr{V}_i)$ zawierająca:

- skończony, niepusty zbiór *lokalnych stanów* $ELoc_i \triangleq Loc_i \times SE$;
- *początkowy, lokalny stan* $e\iota_i \in ELoc_i$;
- skończony, niepusty zbiór *lokalnych akcji* $Act_i = \{a_i^1, a_i^2, \ldots, a_i^{m_i}\}$;
- *lokalny protokół* $EP_i : ELoc_i \to 2^{Act_i} \setminus \{\emptyset\}$;
- (częściową) *lokalną funkcję przejścia* $ET_i : ELoc_i \times Act \to ELoc_i$, zgodną z protokołem, tzn. jeśli $ET_i(e\ell_i, \alpha)$ jest określona, to $\alpha^i \in EP_i(e\ell_i)$;
- skończony, niepusty zbiór *lokalnych zmiennych atomowych* $EAP_i \triangleq AP_i \cup AP_i^E$, gdzie $AP_i^E$ jest zbiorem nowych zmiennych;
- lokalną funkcję wartościującą $\mathscr{E}\mathscr{V}_i : ELoc_i \to 2^{EAP_i}$.

Rozszerzenie to polega na wzbogaceniu lokalnych stanów agentów o wektory opisujące intensywność poszczególnych emocji. Zestaw akcji pozostaje niezmieniony, jednak decyzja o ich wykonaniu może być uzależniona nie tylko od klasycznych warunków, lecz także od aktualnego stanu emocjonalnego agenta, co znajduje odzwierciedlenie w definicji funkcji protokołu. Zgodnie z rozszerzoną funkcją przejścia akcje mogą wpływać na zmianę poziomu emocji, modyfikując tym samym afektywny komponent stanu. Wprowadzamy również dodatkowy zbiór zmiennych atomowych $AP_i^E$, umożliwiający formułowanie własności logicznych odnoszących się do poziomu emocji u poszczególnych agentów. **Afektywny system wieloagentowy** jest to układ złożony z agentów rozszerzonych o komponent afektywny. Formalny **model** takiego systemu budujemy analogicznie do klasycznego modelu MAS, uwzględniając jednak dodatkowe struktury opisujące emocje i ich ewolucję.

### 2.4.3. Strategie afektywne w ATL

Wykorzystując logikę ATL interpretowaną w modelu systemu wieloagentowego rozszerzonego o komponent emocjonalny agentów, możemy definiować własności dotyczące strategii afektywnych. Pozwala to na precyzyjne konstruowanie specyfikacji systemów, w których emocje wpływają na zachowanie agentów. Tak sformułowane

specyfikacje stanowią punkt wyjścia do procesu syntezy, czyli automatycznego generowania modeli spełniających określone własności afektywne.

Poniżej przedstawione zostały cztery przykładowe formuły logiki ATL wyrażające własności systemów wieloagentowych, w których strategie agentów mają wpływ na ich stany emocjonalne.

- **Trwała radość agenta:** *Agent i może tak działać, że od pewnego momentu jego poziom radości będzie zawsze wysoki (+).*

$$\langle\langle\{i\}\rangle\rangle F\langle\langle\{i\}\rangle\rangle G(joy_i^+)$$

- **Unikanie gniewu u agenta:** *Grupa agentów $\{i, j\}$ ma strategię, która zawsze zapobiega wzrostowi gniewu u agenta $j$.*

$$\langle\langle\{i, j\}\rangle\rangle G(\neg anger_j^+)$$

- **Sekwencja radości i smutku u agenta:** *Istnieje strategia agenta i, która prowadzi najpierw do wysokiego poziomu radości, a następnie do wysokiego poziomu smutku.*

$$\langle\langle\{i\}\rangle\rangle F(joy_i^+ \wedge \langle\langle\{i\}\rangle\rangle F(sadness_i^+)$$

- **Stabilność emocjonalna koalicji $\Gamma$:** *Koalicja $\Gamma$ może zagwarantować, że poziom wszystkich emocji wszystkich koalicjantów pozostanie neutralny (0).*

$$\langle\langle\Gamma\rangle\rangle G(\bigwedge_{i\in\Gamma}(joy_i^0 \wedge anger_i^0 \wedge fear_i^0 \wedge disgust_i^0 \wedge sadness_i^0))$$

## 2.5. Formalna metoda konstrukcji modeli specyfikowanych w logice ATL

Model systemu wieloagentowego spełniający specyfikację zadaną formułami logiki ATL może zostać wygenerowany metodą zaimplementowaną w narzędziu MsATL Niewiadomski i in. (2020), której podstawy teoretyczne, wraz ze szczegółami technicznymi i dowodami formalnymi, przedstawiono w pracy Kacprzak, Niewiadomski i Penczek (2020). W niniejszym rozdziale, odwołując się do wspomnianej pracy, cytujemy kluczowe definicje oraz omawiamy podstawowe idee tej metody.

Głównym celem jest znalezienie modelu spełniającego zadaną formułę $\varphi$ lub wykazanie, że taki model nie istnieje w określonej klasie modeli. Model jest reprezentowany symbolicznie jako zbiór zmiennych, które kodują stany globalne, funkcję przejścia oraz funkcję wartościującą. Każde przypisanie wartości boolowskich tym zmiennym koduje inny model.

Proces konstrukcji modelu rozpoczyna się od częściowego przypisania wartości wybranym zmiennym i zdefiniowania ograniczeń opisujących daną klasę modeli.

Następnie algorytm stopniowo rozszerza to przypisanie, aż do uzyskania pełnej funkcji wartościującej. Jeżeli tak skonstruowany model nie spełnia formuły $\varphi$, algorytm cofa się i wypróbowuje inne przypisania.

Poszukiwanie wartościowania spełniającego realizowane jest przez tzw. leniwy weryfikator SMT (ang. *SAT Modulo Theory*), który łączy klasyczny weryfikator SAT z odpowiednimi weryfikatorami teorii. Typowe weryfikatory SMT działają efektywnie dla w pełni określonych danych wejściowych, jednak ich skuteczność spada w przypadku danych częściowych. W odróżnieniu od nich weryfikatory typu SMMT (ang. *SAT Modulo Monotonic Theories*) Bayless, Bayless, Hoos i Hu (2015); Klenze, Bayless i Hu (2016), które są stosowane dla teorii monotonicznych, w przypadku danych częściowych umożliwiają działanie na całych klasach modeli i w konsekwencji szybką redukcję dużej klasy modeli niespełniających formuły $\varphi$. Powoduje to wzrost efektywności zastosowanej metody.

Po uzyskaniu pełnego wartościowania kodującego stany, przejścia i funkcję wartościującą modelu konieczne jest sprawdzenie, czy finalny model spełnia formułę $\varphi$. W tym celu dla logiki ATL z pełną informacją wykorzystuje się narzędzie MCMAS Lomuscio, Qu i Raimondi (2017), natomiast dla ATL z niepełną informacją stosowane jest narzędzie STV Jamroga, Knapik, Kurpiewski i Mikulski (2019); Kurpiewski, Jamroga i Knapik (2019).

### 2.5.1. Kodowanie systemu wieloagentowego i jego modelu

W przyjętym podejściu system wieloagentowy reprezentowany jest za pomocą zbioru nowych zmiennych zdaniowych, niezależnych od tych występujących w języku logiki ATL. Przypisanie tym zmiennym wartości boolowskich pozwala na zakodowanie konkretnych modeli systemu. Przede wszystkim dla każdego agenta $i \in \mathscr{A}gt$ definiujemy trzy zbiory zmiennych zdaniowych: $PB_i$, $TB_i$ oraz $VB_i$, odpowiadające kolejno funkcji protokołu, funkcji przejścia oraz funkcji wartościującej. Niech $\mathscr{PVB} = \bigcup_{i \in \mathscr{A}gt}(TB_i \cup PB_i \cup VB_i)$ oznacza zbiór wszystkich zmiennych wykorzystywanych do opisu modelu. Na tym zbiorze definiujemy funkcję boolowską $v_{\mathscr{M}} : \mathscr{PVB} \to \{0,1\}$, która przypisuje wartości poszczególnym zmiennym, jednoznacznie kodując tym samym dany model systemu. Dokładniej wartości zmiennych ze zbioru:

- $PB_i = \{pb_i(k,t) \mid 1 \leq k \leq |Loc_i|, 1 \leq t \leq |Act_i|\}$ kodują protokół agenta $i$, tzn. **(C1)** $v_{\mathscr{M}}(pb_i(k,t)) = 1$, wtedy i tylko wtedy, gdy akcja $\alpha_i^t$ agenta $i$ jest dozwolona w stanie $\ell_i^k$, tj. $\alpha_i^t \in P_i(l_i^k)$;
- $TB_i = \{tb_i(k,j,k') \mid 1 \leq k,k' \leq |Loc_i|, 1 \leq j \leq |Act|\}$ kodują tranzycje agenta $i$, tzn. **(C2)** $v_{\mathscr{M}}(tb_i(k,j,k')) = 1$, wtedy i tylko wtedy, gdy $T_i(l_i^k, \alpha_j) = l_i^{k'}$;

- $VB_i = \{vb_i(k,j) \mid 1 \leq k \leq |Loc_i|,\ 1 \leq j \leq |AP_i|\}$ kodują wartościowanie zmiennych atomowych agenta $i$, tzn. **(C3)** $v_{\mathscr{M}}(vb_i(k,j)) = 1$, wtedy i tylko wtedy, gdy $\mathsf{p}_i^j$ jest prawdziwa w stanie $\ell_i^k$.

Na zbiorze zmiennych $\mathscr{PVB}$ definiujemy formuły logiczne, które służą do zakodowania zarówno struktury systemu wieloagentowego, jak i jego konkretnego modelu. Konstrukcja tych formuł zależy od przyjętego typu systemu, na przykład czy jest on synchroniczny, czy asynchroniczny, oraz od specyficznych wymagań, jakie powinien spełniać dany system. W szczególności kodowane są następujące właściwości MAS:

- **A1** - w każdym stanie lokalnym jest co najmniej jedna dozwolona akcja

$$\varphi_1 = \bigwedge_{i \leq n; k \leq n_i} \bigvee_{t \leq m_i} pb_i(k,t);$$

- **A2** - tranzycje lokalne przebiegają tylko po akcjach dozwolonych protokołem

$$\varphi_2 = \bigwedge_{i \leq n; t \leq m_i; k \leq n_i; j \in gl_i(t)} \left( \left( \bigvee_{k' \leq n_i} tb_i(k,j,k') \right) \leftrightarrow pb_i(k,t) \right);$$

- **A3** - relacja przejścia jest funkcją, czyli akcje są deterministyczne

$$\varphi_3 = \bigwedge_{i \leq n; k,k' \leq n_i; j \leq |Act|} \left( tb_i(k,j,k') \rightarrow \bigwedge_{k'' \leq n_i, k'' \neq k'} \neg tb_i(k,j,k'') \right).$$

Podobnie definiujemy warunki kodujące model systemu, tzn. jego:

- tranzycje globalne: $T(q,a) = q'$ wttw $\forall_{i \leq n} v_{\mathscr{M}}(tb_i(q_i,a,q_i')) = 1$,
- wartościowanie zmiennych atomowych w stanach globalnych: $p_i^j \in \mathscr{V}(q)$ wttw $v_{\mathscr{M}}(vb_i(q_i,j)) = 1$.

W konsekwencji budowanie modelu dla formuły $\psi$ sprowadza się do wyznaczenia funkcji $v_{\mathscr{M}}$ spełniającej zadane wymagania i formuły.

### 2.5.2. Model częściowy i jego dopełnienia

Jak już wcześniej wspomniano, model systemu wieloagentowego reprezentowany jest poprzez wartości boolowskie przypisane nowo wprowadzonym zmiennym zdaniowym ze zbioru $\mathscr{PVB}$. Na początku procedury syntezy taki model nie istnieje lub, jeśli podano warunki początkowe bądź dodatkowe wymagania, jest jedynie częściowo skonstruowany.

Sformalizowanie tej niepełności wymaga rozszerzenia zbioru wartości boolowskich do zbioru trójwartościowego poprzez wprowadzenie nowej wartości $u$, która oznacza zmienną *nieokreśloną*. W ten sposób uzyskujemy pojęcia częściowych agentów, częściowego systemu wieloagentowego oraz częściowego modelu. Oznacza to również, że globalna funkcja przejścia oraz wartościowanie zmiennych w stanach globalnych są zdefiniowane tylko częściowo.

Niech $B_u = \{0,1,u\}$. *Częściowy model* $\mathscr{M}_P$ jest zakodowany poprzez funkcję wartościowania:

$$v_{\mathscr{M}_P} \colon \mathscr{PVB} \to B_u,$$

która przypisuje wartość $u$ wszystkim zmiennym niezdecydowanym. Celem algorytmu syntezy jest rozszerzenie funkcji $v_{\mathscr{M}_P}$ do totalnej funkcji o wartościach boolowskich 0,1, spełniającej warunki **C1–C3** oraz formuły **A1–A3**.

Dla danego modelu częściowego $\mathscr{M}_P$ oraz grupy agentów $\Gamma$ definiujemy dwa totalne modele, zwane odpowiednio *dopełnieniem dolnym* i *dopełnieniem górnym*.

*Dopełnienie górne* $\mathscr{M}^{\Gamma}_{\text{over}}$ modelu częściowego $\mathscr{M}_P$ względem grupy $\Gamma$ polega na przypisaniu wartości 1 wszystkim niezdecydowanym zmiennym: kodującym wartościowanie zmiennych atomowych wszystkich agentów oraz kodującym funkcje protokołu i przejścia (tranzycji) agentów należących do grupy $\Gamma$. Jednocześnie wszystkim niezdecydowanym wartościom kodującym funkcje protokołu i tranzycji agentów spoza grupy $\Gamma$ (tj. z $\mathscr{A}gt \setminus \Gamma$) przypisywana jest wartość 0.

Analogicznie, *dopełnienie dolne* $\mathscr{M}^{\Gamma}_{\text{under}}$ względem grupy $\Gamma$ przypisuje wartość 0 wszystkim niezdecydowanym zmiennym: kodującym wartościowanie zmiennych atomowych wszystkich agentów, kodującym funkcje protokołu i tranzycji agentów z grupy $\Gamma$, natomiast wszystkim niezdecydowanym wartościom odnoszącym się do funkcji protokołu i tranzycji agentów z $\mathscr{A}gt \setminus \Gamma$ przypisywana jest wartość 1.

Zauważmy, że dopełnienie dolne modelu względem grupy agentów $\Gamma$ maksymalnie ogranicza możliwe działania i przejścia agentów należących do tej grupy, przy jednoczesnym dopuszczeniu wszystkich możliwych ruchów dla agentów spoza $\Gamma$. Z kolei dopełnienie górne działa odwrotnie – agenci z grupy $\Gamma$ mogą wykonać wszystkie możliwe ruchy, natomiast działania agentów spoza tej grupy są ściśle ograniczone.

Ponadto dla modelu częściowego wprowadzamy pojęcie *rozszerzenia*. Mówimy, że model $\mathscr{M}$ *rozszerza* model $\mathscr{M}'$, jeśli wartości przypisane przez kodowanie tych modeli są zgodne dla wszystkich zmiennych, które mają wartość określoną w $\mathscr{M}'$. Dla zmiennych nieokreślonych w $\mathscr{M}'$ model $\mathscr{M}$ może przypisać dowolne wartości.

Wówczas dla $\Gamma \subseteq \mathscr{A}gt$ prawdziwe są zależności:

- $v_{(\mathscr{M}')^{\Gamma}_{under}}(b_i) \leq v_{\mathscr{M}^{\Gamma}_{under}}(b_i) \leq v_{\mathscr{M}^{\Gamma}_{over}}(b_i) \leq v_{(\mathscr{M}')^{\Gamma}_{over}}(b_i)$ dla $b_i \in TB_i \cup PB_i$ i $i \in \Gamma$;
- $v_{(\mathscr{M}')^{\Gamma}_{under}}(b_i) \geq v_{\mathscr{M}^{\Gamma}_{under}}(b_i) \geq v_{\mathscr{M}^{\Gamma}_{over}}(b_i) \geq v_{(\mathscr{M}')^{\Gamma}_{over}}(b_i)$ dla $b_i \in TB_i \cup PB_i$ i $i \notin \Gamma$;
- $v_{(\mathscr{M}')^{\Gamma}_{under}}(vb) \leq v_{\mathscr{M}^{\Gamma}_{under}}(vb) \leq v_{\mathscr{M}^{\Gamma}_{over}}(vb) \leq v_{(\mathscr{M}')^{\Gamma}_{over}}(vb)$ dla $vb \in VB$.

Zbiór wszystkich możliwych pełnych modeli można uzyskać poprzez uzupełnienie nieokreślonych wartości w modelu częściowym w sposób zgodny z warunkami

poprawności. W rezultacie model częściowy wyznacza klasę dopuszczalnych rozszerzeń, które mogą następnie zostać poddane analizie pod kątem spełnialności zadanej formuły. Takie podejście umożliwia iteracyjną konstrukcję pełnych modeli przy jednoczesnym eliminowaniu tych fragmentów przestrzeni rozwiązań, które nie prowadzą do spełnienia wymagań.

### 2.5.3. Monotoniczność

Algorytm spełnialności i syntezy, który przedstawiamy, opiera się na technice stosowanej wcześniej w kontekście teorii monotonicznych dla sprawdzania spełnialności formuł w logice CTL Bayless i in. (2015); Klenze i in. (2016). Metoda ta została następnie rozszerzona do logiki ATL Kacprzak i in. (2020); Niewiadomski i in. (2020) oraz do spełnialności formuł w logice strategicznej SL[SG] Kacprzak, Niewiadomski i Penczek (2021).

Boolowska teoria monotoniczna (ang. *Boolean Monotonic Theory*) to klasa teorii logicznych, których modele mają pewną właściwość monotoniczności względem wartościowania zmiennych boolowskich.

Niech $P\colon B^n \to B$ będzie predykatem. Mówimy, że predykat $P$ jest *boolowsko monotoniczny pozytywnie (dodatnio)*, jeśli dla każdego $1 \leq i \leq n$ zachodzi: $P(s_1,\ldots,s_{i-1},0,s_{i+1},\ldots,s_n) \leq P(s_1,\ldots,s_{i-1},1,s_{i+1},\ldots,s_n)$. Analogicznie predykat $P$ jest *boolowsko monotoniczny negatywnie (ujemnie)*, jeśli dla każdego $i$: $P(s_1,\ldots,s_{i-1},1,s_{i+1},\ldots,s_n) \leq P(s_1,\ldots,s_{i-1},0,s_{i+1},\ldots,s_n)$. Powyższe definicje rozszerzają się naturalnie na funkcje $F\colon B^n \to 2^S$, gdzie $S$ jest dowolnym zbiorem.

**Definicja 2.8 (Boolowska teoria monotoniczna).** Teoria $T$ o sygnaturze $\Omega = (S, S_f, S_r, Ar)$, gdzie $S$ to niepusty zbiór elementów dziedziny, $S_f$ – zbiór symboli funkcji, $S_r$ – zbiór symboli relacji, a $Ar$ określa arność funkcji i relacji, nazywa się *boolowsko monotoniczną*, jeśli wszystkie elementy dziedziny w $\Omega$ są boolowskie, a wszystkie funkcje i predykaty w $\Omega$ są monotoniczne (w rozumieniu powyższych definicji).

Relacja spełniania logiki ATL, zarówno dla semantyki z wiedzą pełną jak i niepełną, nie jest monotoniczna, ale spełnia warunek monotoniczności w pewnych szczególnych przypadkach, co opisuje poniższe twierdzenie udowodnione w Kacprzak i in. (2020).

**Twierdzenie 2.1.** *Niech $\mathscr{M}$ i $\mathscr{M}'$ będą dwoma modelami zdefiniowanymi dla tego samego zbioru agentów $\mathscr{A}gt$, dla dwóch* MAS*, które dzielą te same zbiory $Loc_i$, $Act_i$ oraz $AP_i$ dla każdego $i \in \mathscr{A}gt$, oraz niech $Y \in \{Ir, ir\}$. Wówczas:*

$$\mathscr{M}, q \models_Y \phi \Rightarrow \mathscr{M}', q \models_Y \phi,$$

*jeśli spełnione są następujące warunki:*

*(HA1)* $\phi \in \{\mathsf{p}, \mathsf{p}_1 \wedge \mathsf{p}_2\}$ *oraz* $v_{\mathscr{M}}(vb_i) \leq v_{\mathscr{M}'}(vb_i)$ *dla każdego* $vb_i \in VB_i$, $i \in \mathscr{A}gt$,
*(HA2)* $\phi = \neg\mathsf{p}$ *oraz* $v_{\mathscr{M}}(vb_i) \geq v_{\mathscr{M}'}(vb_i)$ *dla każdego* $vb_i \in VB_i$, $i \in \mathscr{A}gt$,
*(HA3)* $\phi \in \{\langle\langle\Gamma\rangle\rangle X\mathsf{p}, \langle\langle\Gamma\rangle\rangle G\mathsf{p}, \langle\langle\Gamma\rangle\rangle \mathsf{p}_1 U \mathsf{p}_2\}$ *oraz*

*(HA3.1)* $v_{\mathscr{M}}(vb_i) \leq v_{\mathscr{M}'}(vb_i)$ *dla każdego* $vb_i \in VB_i$, $i \in \mathscr{A}gt$,
*(HA3.2)* $v_{\mathscr{M}}(b_i) \leq v_{\mathscr{M}'}(b_i)$ *dla każdego* $b_i \in TB_i \cup PB_i$ *oraz* $i \in \Gamma$,
*(HA3.3)* $v_{\mathscr{M}}(b_i) \geq v_{\mathscr{M}'}(b_i)$ *dla każdego* $b_i \in TB_i \cup PB_i$ *oraz* $i \in \mathscr{A}gt \setminus \Gamma$.

Przypadek **(HA2)** wykazuje *negatywną monotoniczność* operatora negacji względem zmiennych kodujących zmienne atomowe. Z kolei przypadek **(HA3.3)** potwierdza *negatywną monotoniczność* operatora strategicznego względem zmiennych kodujących tranzycje kontrolowane przez agentów spoza koalicji $\Gamma$. We wszystkich pozostałych przypadkach zachodzi *pozytywna monotoniczność*. Należy zauważyć, że wszystkie te przypadki odnoszą się do formuł niezagnieżdżonych. Powyższe własności implikują wymaganą monotoniczność w procedurze syntezy dla danej logiki.

### 2.5.4. Aproksymacje modeli częściowych

Dla danego modelu częściowego $\mathscr{M}'$ i formuły $\phi$ wyznaczane są dwa przybliżenia:

- **przybliżenie modeli** – zbiór modeli oznaczony jako $CLASS_{\mathscr{M}'}(\phi)$, taki że jeśli $\mathscr{M}$ jest całkowitym rozszerzeniem $\mathscr{M}'$ i $\mathscr{M}, q \models_Y \phi$ dla pewnego $q \in \mathscr{Q}$, to $\mathscr{M} \in CLASS_{\mathscr{M}'}(\phi)$;
- **przybliżenie stanów** – zbiór stanów oznaczony przez $\|\phi\|_{\mathscr{M}'}$, zawierający wszystkie stany spełniające $\phi$ w jakimś modelu z $CLASS_{\mathscr{M}'}(\phi)$, tj. $\{q \in \mathscr{Q} \mid \exists \mathscr{M} \in CLASS_{\mathscr{M}'}(\phi)$, taki że $\mathscr{M}, q \models_Y \phi\} \subseteq \|\phi\|_{\mathscr{M}'}$.

Ponieważ $CLASS_{\mathscr{M}'}(\phi)$ i $\|\phi\|_{\mathscr{M}'}$ są tylko przybliżeniami, modele i stany, które nie spełniają formuły, również mogą być w nich zawarte. Dodatkowo dla ustalonego modelu częściowego $\mathscr{M}'$, formuły $\phi$ i początkowego stanu $\iota$ z definicji wynika, że jeśli $\mathscr{M}'$ może być rozszerzony do modelu pełnego $\mathscr{M}$, takiego że $\mathscr{M}, \iota \models_Y \phi$, to:

$$\mathscr{M} \in CLASS_{\mathscr{M}'}(\phi) \text{ oraz } \iota \in \|\phi\|_{\mathscr{M}'}.$$

Przybliżenia wykorzystywane są w procedurze sprawdzania spełnialności do wczesnego wykrywania konfliktów, co umożliwia szybsze znajdowanie modelu. Jeśli ustalony stan początkowy nie należy do wyznaczonego przybliżenia stanów, to klasa modeli określona tym przybliżeniem może zostać odrzucona, co znacznie zwiększa efektywność procedury.

Dla modelu totalnego $\mathscr{M}$ rozszerzającego model częściowy $\mathscr{M}'$ algorytm 2.1 o nazwie *Apx* oblicza nad- i podaproksymację zbioru stanów $\|\phi\|_{\mathscr{M}}$, a mianowicie $Apx(\phi, \mathscr{M}', over)$ zwraca zbiór stanów będący nadaproksymacją zbioru $\|\phi\|_{\mathscr{M}}$. Oznacza to, że jeśli $\iota \in \|\phi\|_{\mathscr{M}}$, to $\iota \in Apx(\phi, \mathscr{M}', over)$. Zatem:

**Algorytm 2.1** Algorytm $Apx$

**Input**: $\phi, \mathscr{M}$ dla częściowego MAS, $\lambda \in \{over, under\}$
**Output**: zbiór stanów modelu $\mathscr{M}$

**if** $\phi \in AP$ **then**
  **return** $\{q \in \mathscr{Q} \mid \mathscr{M}_\lambda^{\mathscr{A}gt}, q \models_{\mathscr{L}} \phi\}$
**else if** $\phi = op(\psi)$ **then**
  **if** $op = \neg$ **then** // negatywna monotoniczność
    $Z := Apx(\psi, \mathscr{M}, \overline{\lambda})$
    **return** $\mathscr{Q} \setminus Z$
  **else** // op $\in \{\langle\langle\Gamma\rangle\rangle \mathrm{X}, \langle\langle\Gamma\rangle\rangle \mathrm{G}\}$
    $Z := Apx(\psi, \mathscr{M}, \lambda)$
    **return** $MC(op, Z, v_{M_\lambda^\Gamma})$
  **end if**
**else if** $\phi = op(\psi_1, \psi_2)$ dla $op \in \{\langle\langle\Gamma\rangle\rangle U, \wedge\}$ **then**
  $Z_1 := Apx(\psi_1, \mathscr{M}, \lambda)$
  $Z_2 := Apx(\psi_2, \mathscr{M}, \lambda)$
  **if** $op$ jest $\langle\langle\Gamma\rangle\rangle U$ **then**
    **return** $MC(op, Z_1, Z_2, v_{M_\lambda^\Gamma})$
  **else** // op $= \wedge$
    **return** $Z_1 \cap Z_2$
  **end if**
**end if**

jeśli $\iota \notin Apx(\phi, \mathscr{M}', over)$,
to nie istnieje model $\mathscr{M}$ rozszerzający $\mathscr{M}'$, taki że $\mathscr{M}, \iota \models_Y \phi$.

Analogicznie $Apx(\phi, \mathscr{M}', under)$ oblicza podaproksymację zbioru $\|\phi\|_{\mathscr{M}}$, tzn. jeśli $\iota \in Apx(\phi, \mathscr{M}', under)$, to $\iota \in \|\phi\|_{\mathscr{M}}$. Własności te są podstawą działania procedury syntezy. Ich dowody przedstawione zostały w pracy Kacprzak i in. (2020).

Poprawność algorytmu 2.1 wynika z własności monotoniczności użytej w nim procedury weryfikacji modelowej $MC$ Kacprzak i in. (2020). Funkcja $MC(op, Z_1, v_{\mathscr{M}})$ jest zdefiniowana dla operatora unarnego $op$, natomiast $MC(op, Z_1, Z_2, v_{\mathscr{M}})$ dla operatora binarnego $op$, gdzie $Z_1, Z_2 \subseteq \mathscr{Q}$, a $v_{\mathscr{M}}$ to wartościowanie kodujące model $\mathscr{M}$. $MC$ oblicza operator $op$ na zbiorach stanów. Jeśli $\phi = \mathsf{p} \in AP$, to funkcja zwraca zbiór stanów modelu $\mathscr{M}$, w których $\mathsf{p}$ jest spełnione, tzn. $\{q \in \mathscr{Q} \mid \mathsf{p} \in \mathscr{V}(q)\}$. W przeciwnym razie pobiera najwyższy operator $op$ z $\phi$ i rozwiązuje jego argument(y) rekurencyjnie, stosując $MC(op, Z_1, v_{\mathscr{M}})$ lub $MC(op, Z_1, Z_2, v_{\mathscr{M}})$ do uzyskanych zbiorów stanów. Zauważmy, że $MC(\neg, Z, v_{\mathscr{M}})$ zwraca $\mathscr{Q} \setminus Z$ oraz $MC(\wedge, Z_1, Z_2, v_{\mathscr{M}})$ zwraca $Z_1 \cap Z_2$. Jeśli $op \in \{\langle\langle\Gamma\rangle\rangle X\,\mathsf{p}, \langle\langle\Gamma\rangle\rangle G\,\mathsf{p}, \langle\langle\Gamma\rangle\rangle \mathsf{p}_1 U \mathsf{p}_2\}$, to stosowany jest algorytm weryfikacji modelowej dla logiki ATL Kurpiewski i in. (2019); Lomuscio i in. (2017).

### 2.5.5. Procedura spełnialności i syntezy

Procedurę syntezy i tym samym badania spełnialności dla zadanej formuły $\phi$ zaczynamy od ustalenia początkowych wymagań dotyczących modelu, tzn. liczby jego agentów, liczby lokalnych akcji dla każdego agenta, liczby lokalnych stanów dla każdego agenta, liczby lokalnych zmiennych atomowych dla każdego agenta oraz dodatkowych warunków, takich jak wymagania **A1 – A3**. Zdefiniowane wymagania, wraz z formułą $\phi$ oraz wyznaczonym stanem początkowym $\iota$, stanowią dane wejściowe dla algorytmu konstrukcji modelu. Oczekiwanym wynikiem działania algorytmu jest model totalny $\mathscr{M}$, który spełnia wszystkie zadane ograniczenia oraz $\mathscr{M}, \iota \models_Y \phi$. W przypadku braku takiego modelu algorytm zwraca informację o jego nieistnieniu.

Niech $depth$ będzie zmienną o wartościach całkowitych śledzącą głębokość decyzji weryfikatora. Poniżej opisane zostały poszczególne etapy procedury.

Kolejne kroki algorytmu:

1. Ustaw $depth := 0$.
   Wyznacz częściową funkcję wartościującą $v_{\mathscr{M}}$ zgodnie z zadanymi wymaganiami początkowymi. Sprawdź, czy $v_{\mathscr{M}}$ spełnia warunki **A1–A3**:
   - jeśli tak – przejdź do kroku 2,
   - jeśli nie – zwróć **UNSAT**.
2. Oblicz $Apx(\phi, \mathscr{M}, over)$.
3. Jeśli $\iota \in Apx(\phi, \mathscr{M}, over)$, to:
   a. jeśli model $\mathscr{M}$ jest totalny, zwróć **SAT** oraz skonstruowany model $\mathscr{M}$;
   b. w przeciwnym razie:
      - zwiększ głębokość: $depth := depth + 1$;
      - przypisz wartość nieustalonej zmiennej w $v_{\mathscr{M}}$ zgodnie z warunkami **A1–A3**;
      - zaktualizuj model częściowy $\mathscr{M}$, zawężając rozważaną klasę modeli;
      - przejdź do kroku 2.
4. Jeśli $\iota \notin Apx(\phi, \mathscr{M}, over)$, to:
   a. jeśli $depth > 0$:
      - przeanalizuj konflikt;
      - cofnij decyzje do głębokości konfliktu $c$;
      - przypisz przeciwną wartość konfliktowej zmiennej;
      - ustaw $depth := c$ i zaktualizuj model częściowy $\mathscr{M}$;
      - przejdź do kroku 2.

b. jeśli $depth = 0$, zwróć **UNSAT**.

Główna idea algorytmu polega na stopniowej konstrukcji modelu poprzez sukcesywne przypisywanie wartości zmiennym boolowskim, które go kodują. Wartości te częściowo wynikają z początkowych ograniczeń, co pozwala zdefiniować początkową klasę dopuszczalnych modeli. Wykorzystując własność monotoniczności, algorytm sprawdza, czy dany model częściowy można rozszerzyć do modelu totalnego spełniającego formułę $\phi$. W przypadku braku konfliktów następuje dalsze zawężanie klasy modeli; w przeciwnym razie algorytm cofa się do wcześniejszego etapu i modyfikuje decyzje. Procedura kończy się, gdy zostanie znaleziony model spełniający $\phi$ lub gdy żadna możliwa klasa modeli nie pozostaje do rozważenia. Jeśli taki model istnieje, końcowa aproksymacja będzie jednoznacznie go wyznaczać, natomiast zidentyfikowane stany będą tymi, w których formuła $\phi$ jest spełniona.

## 2.6. Generowanie modeli afektywnych systemów wieloagentowych

Omówiona w poprzednim rozdziale metoda może zostać zastosowana do syntezy modeli afektywnych systemów wieloagentowych, jeśli ich specyfikacja jest wyrażona w języku logiki ATL. Formalizm ten okazuje się wystarczająco elastyczny, aby umożliwić opis strategii afektywnych, o ile zbiór zmiennych zdaniowych, nad którymi budowane są formuły tej logiki, zostanie rozszerzony o zmienne reprezentujące stany emocjonalne agentów:

$$AP_i^E = \{\mathsf{ep}_i^j \ : \ j = 1,2,\ldots,k\}$$

dla $i \in \mathscr{A}gt$ oraz pewnego $k \in \mathbb{N}$. Są to na przykład takie zmienne jak $joy^+$ (wysoki poziom zadowolenia) czy $sadness^-$ (brak smutku).

Zmienne te mogą być interpretowane jako wskaźniki natężenia emocji w danym stanie lokalnym i służą do specyfikacji właściwości afektywnych analizowanego systemu. Ich wprowadzenie nie narusza ani składni, ani semantyki logiki ATL, a jedynie wzbogaca przestrzeń interpretacji o dodatkowe atrybuty emocjonalne. W konsekwencji proponowane rozszerzenie zachowuje pełną zgodność z klasycznym aparatem formalnym logiki ATL, jednocześnie umożliwiając jej zastosowanie w modelowaniu i analizie systemów kognitywnych oraz adaptacyjnych, w których emocje odgrywają istotną rolę w procesach decyzyjnych agentów.

### 2.6.1. Innowacyjność rozwiązania

Przedstawione w niniejszej pracy podejście stanowi rozszerzenie klasycznych metod weryfikacji i syntezy modeli w logice ATL o komponent afektywny, umożliwiający modelowanie i analizę systemów, w których agenci dysponują nie tylko zestawem strategii racjonalnych, lecz także zdolnością do odczuwania, rejestrowania i modyfikowania stanów emocjonalnych. Nowość tej koncepcji polega na wprowadzeniu do lokalnych stanów agentów wektorów emocjonalnych, które opisują natężenie i dynamikę wybranych emocji, takich jak strach, radość czy złość. Takie ujęcie pozwala na badanie wpływu emocji na decyzje strategiczne agentów, a tym samym na modelowanie bardziej realistycznych, kognitywnych zachowań w systemach wieloagentowych.

Wkład pracy ma charakter zarówno teoretyczny, jak i metodologiczny. Jeśli chodzi o stronę teoretyczną, to zaproponowano formalne rozszerzenie modelu MAS o komponent afektywny, przy zachowaniu zgodności z klasyczną semantyką logiki ATL. W zakresie metodologii – przedstawiono sposób konstrukcji modeli spełniających formuły logiki ATL z komponentem afektywnym, wykorzystując istniejącą metodę syntezy modeli, wcześniej stosowaną do weryfikacji teorii monotonicznych. Umożliwia to automatyczne generowanie modeli afektywnych, które spełniają określone specyfikacje logiczne.

### 2.6.2. Trudność uogólnienia metody

Uogólnienie metody zaprojektowanej pierwotnie dla klasycznych systemów dla logiki ATL do systemów afektywnych napotyka szereg wyzwań. Najistotniejszym z nich jest potrzeba rozszerzenia przestrzeni stanów lokalnych agentów o dodatkowe wymiary opisujące ich emocje, przy jednoczesnym zachowaniu spójności semantycznej modelu. W klasycznym podejściu stany lokalne opisują możliwe sytuacje, w których może znajdować się agent w wyniku swoich obserwacji i interakcji ze środowiskiem. W modelu afektywnym muszą one dodatkowo uwzględniać zmienne opisujące natężenie emocji oraz ich ewolucję w czasie. Wymaga to przedefiniowania funkcji protokołu i funkcji przejścia w taki sposób, aby akcje mogły wpływać nie tylko na aspekty poznawcze, lecz także na stan emocjonalny agenta.

Z formalnego punktu widzenia trudność polega na zachowaniu monotoniczności teorii po dodaniu wymiaru afektywnego. Warunkiem zachowania stosowalności techniki SMMT jest zapewnienie, że rozszerzenie nie prowadzi do powstania nowych zależności nieliniowych naruszających monotoniczność formuł. Odpowiednie uogólnienie wymaga zatem ostrożnego zaprojektowania struktury stanów emocjonalnych i ich interakcji z klasycznymi zmiennymi logicznymi.

### 2.6.3. Zachowanie właściwości monotoniczności

Jednym z kluczowych elementów opisywanej metody syntezy jest boolowskie kodowanie konstruowanego modelu, a co za tym idzie, wyznaczenie wartości zmiennych boolowskich reprezentujących dany model. Do tego celu zdefiniowane są trzy zbiory zmiennych zdaniowych: $TB_i$, $PB_i$, $VB_i$ dla $i \in \mathscr{A}gt$, kodujące odpowiednio: protokół, tranzycje i wartościownaie zmiennych atomowych w stanach lokalnych. Po wprowadzeniu do języka logiki ATL zmiennych reprezentujących natężenie emocji agentów nowe zmienne muszą reprezentować również zmienne związane z emocjami. Dlatego w zbiorze $VB_i$ wyodrębniamy podzbiór

$$VB_i \supseteq EVB_i = \{evb_i(k,j) \mid 1 \leq k \leq |ELoc_i|,\ 1 \leq j \leq |AP_i^E|\}$$

zmiennych, które kodują wartościowanie zmiennych afektywnych agenta $i$, tzn. $v_{\mathscr{M}}(evb_i(k,j)) = 1$ wtedy i tylko wtedy, gdy zmienna $\mathsf{ep}_i^j$ kodująca natężenie emocji jest prawdziwa w stanie lokalnym $e\ell_i^k$. Kodowanie funkcji protokołu i tranzycji pozostawiamy bez zmian. Zmienne afektywne zachowują własność pozytywnej monotoniczności.

**Twierdzenie 2.2.** *Niech $\mathscr{M}$ i $\mathscr{M}'$ będą dwoma modelami zdefiniowanymi dla tego samego zbioru agentów $\mathscr{A}gt$, dla dwóch afektywnych systemów wieloagentowych, które dzielą te same zbiory $ELoc_i, Act_i, EAP_i$ dla każdego $i \in \mathscr{A}gt$ oraz niech $Y \in \{Ir, ir\}$. Wówczas dla każdego $i \in \mathscr{A}gt$ oraz $evb_i \in EVB_i$, jeśli*

$$v_{\mathscr{M}}(evb_i) \leq v_{\mathscr{M}'}(evb_i)$$

*to dla każdego* $\mathsf{ep} \in AP_i^E$

$$\mathscr{M}, q \models_Y \mathsf{ep} \Rightarrow \mathscr{M}', q \models_Y \mathsf{ep}.$$

Uzasadnienie prawdziwości tego twierdzenia przeprowadzimy metodą nie wprost. Niech $\mathscr{M}$ i $\mathscr{M}'$ będą modelami spełniającymi założenia twierdzenia, $q$ stanem globalnym oraz $\mathsf{ep} \in AP_i^E$ dla $i \in \mathscr{A}gt$ . Załóżmy, że $\mathscr{M}', q \not\models_Y \mathsf{ep}$. Zatem, z semantyki zmiennych atomowych, $\mathsf{ep} \notin \mathscr{EV}'(q)$ oraz $\mathsf{ep} \notin \mathscr{EV}'_i(q_i)$. Stąd $v_{\mathscr{M}'}(evb_i(q_i, \mathsf{ep})) = 0$. Z założenia $v_{\mathscr{M}}(evb_i(q_i, \mathsf{ep})) = 0$ i w konsekwencji
$\mathsf{ep} \notin \mathscr{EV}_i(q_i)$ oraz $\mathsf{ep} \notin \mathscr{EV}(q)$, czyli $\mathscr{M}, q \not\models_Y \mathsf{ep}$.

Zmienne afektywne umożliwiają wnioskowanie o stanach emocjonalnych agentów, a ich spełnialność zależy od afektywnego rozszerzenia stanów lokalnych, które obejmuje dodatkowe atrybuty opisujące intensywność emocji. Z formalnego punktu widzenia semantyka tych zmiennych jest określona analogicznie do pozostałych zmiennych atomowych, tj. za pomocą funkcji wartościującej. W konsekwencji ich wprowadzenie nie wpływa na fundamentalne własności systemu formalnego, w szczególności nie narusza jego monotoniczności.

Dowody pozytywnej i negatywnej monotoniczności dla nieafektywnych zmiennych atomowych, spójników boolowskich oraz modalności strategicznych i temporalnych pozostają takie same jak dla klasycznej logiki ATL i klasycznego modelu wieloagentowego Kacprzak i in. (2020).

### 2.6.4. Uzasadnienie poprawności metody

Pomimo wprowadzenia komponentu afektywnego, poza dowodem zachowania monotoniczności nie są konieczne nowe dowody poprawności metody. Wynika to z faktu, że rozszerzenie nie modyfikuje podstawowych własności semantycznych logiki ATL – operatory strategii, relacje dostępności i sposób interpretacji formuł pozostają niezmienione. Nowy aspekt dotyczy wyłącznie sposobu reprezentacji stanów agentów, który mieści się w ogólnym schemacie definicyjnym modelu MAS. W konsekwencji wszystkie twierdzenia o poprawności i zupełności metody weryfikacji oraz syntezy formuł ATL zachowują ważność również w rozszerzonym kontekście.

Tym samym zaproponowane podejście można traktować jako rozszerzenie metody opracowanej dla logiki ATL Kacprzak i in. (2020), Niewiadomski i in. (2020) i jej uogólnienia SL Kacprzak i in. (2021), które wzbogaca ją o możliwość modelowania emocji, zachowując pełną zgodność z istniejącą aparaturą formalną i narzędziową.

## 2.7. Przykład

Naszym celem jest opracowanie formalnego modelu systemu złożonego z dwóch współdziałających agentów: **Auto** oraz **Radio**. Agent **Auto** analizuje emocjonalny stan kierowcy na podstawie jego mimiki, tonu głosu i postawy ciała, identyfikując jeden z trzech podstawowych stanów: *irytację*, *opanowanie* lub *znużenie*. Agent **Radio** w odpowiedzi na rozpoznany stan emocjonalny dobiera odpowiednią muzykę – relaksującą, dynamiczną lub popularną – w celu jego modyfikacji. W kolejnych sekcjach przedstawimy specyfikację takiego systemu w logice ATL, a następnie zaproponujemy odpowiadający jej model formalny.

### 2.7.1. Specyfikacja systemu w logice ATL

Poniżej prezentujemy wybrane formuły logiki ATL, które stanowią podstawę formalnej specyfikacji rozważanego systemu. Opisują one zależności między stanami emocjonalnymi kierowcy, strategiami podejmowanymi przez agentów oraz możliwymi przejściami między stanami systemu. W formułach tych wykorzystujemy

trzy zmienne atomowe oznaczające aktualny stan emocjonalny kierowcy: irritated, calm oraz bored, a także trzy zmienne reprezentujące aktualnie odtwarzaną muzykę: relaxing, pop i dynamic.

1. Radio reaguje poprawnie na stan emocjonalny:

$$\langle\langle\{Radio\}\rangle\rangle G\Big(\mathsf{irritated} \to \langle\langle\{Radio\}\rangle\rangle X\,\mathsf{relaxing}\Big),$$
$$\langle\langle\{Radio\}\rangle\rangle G\Big(\mathsf{bored} \to \langle\langle\{Radio\}\rangle\rangle X\,\mathsf{dynamic}\Big),$$
$$\langle\langle\{Radio\}\rangle\rangle G\Big(\mathsf{calm} \to \langle\langle\{Radio\}\rangle\rangle X\,\mathsf{pop}\Big).$$

2. Nie jest możliwe bezpośrednie przejście między irytacją a znużeniem:

$$\langle\langle\{Auto\}\rangle\rangle G\Big(\mathsf{irritated} \to \neg\langle\langle\{Auto\}\rangle\rangle X\,\mathsf{bored}\Big),$$
$$\langle\langle\{Auto\}\rangle\rangle G\Big(\mathsf{bored} \to \neg\langle\langle\{Auto\}\rangle\rangle X\,\mathsf{irritated}\Big).$$

3. Możliwa jest cykliczna stabilizacja emocjonalna:

$$\langle\langle\{Auto, Radio\}\rangle\rangle G\ (\langle\langle\{Auto, Radio\}\rangle\rangle F\,\mathsf{calm}).$$

4. Irytacja zawsze powinna prowadzić do uspokojenia:

$$\langle\langle\{Auto, Radio\}\rangle\rangle G\Big(\mathsf{irritated} \to \langle\langle\{Auto, Radio\}\rangle\rangle F(\neg\mathsf{irritated})\Big).$$

4. Znużenie zawsze powinno prowadzić do pobudzenia:

$$\langle\langle\{Auto, Radio\}\rangle\rangle G\Big(\mathsf{bored} \to \langle\langle\{Auto, Radio\}\rangle\rangle F(\neg\mathsf{bored})\Big).$$

### 2.7.2. Przykładowy system wieloagentowy

Opiszemy teraz system wieloagentowy realizujący powyższą specyfikację.

### Agent Auto

Dla uproszczenia przyjmujemy, że agent Auto może znajdować się wyłącznie w jednym z trzech stanów emocjonalnych, które odzwierciedlają zmienność poziomu złości – od niskiego, przez neutralny, aż po wysoki. Pozostałe emocje pozostają niezmienne w czasie. Stan lokalny agenta ogranicza się jedynie do aspektu afektywnego; nie uwzględniamy żadnych dodatkowych atrybutów. W bardziej zaawansowanej wersji systemu możliwe byłoby jednak rozszerzenie modelu o kolejne komponenty stanu. Zatem:

$$ELoc_{Auto} = \{(0,0,0,0,-1),(0,0,0,0,0),(0,0,0,0,1)\}.$$

Dla ułatwienia nazwiemy te stany tak samo jak zmienne atomowe, które w tych stanach będą prawdziwe:

$$ELoc_{Auto} = \{\mathsf{irritated}, \mathsf{calm}, \mathsf{bored}\}.$$

Agent zaczyna w stanie początkowym: calm.
Dostępne są trzy akcje zmiany oraz jedna akcja oznaczająca brak nowych operacji:

$$Act_{Auto} = \{\texttt{detectCalm}, \texttt{detectIrritation}, \texttt{detectBoredom}, \texttt{noop}\}.$$

Możliwe przejścia między stanami, a zarazem protokół działania agenta Auto, zostały określone za pomocą tranzycji przedstawionych w tabeli 2.1. Symbol $*$ oznacza w nich dowolną akcję podejmowaną przez agenta Radio. Przypomnijmy, że tranzycje są etykietowane akcjami globalnymi, będącymi krotkami decyzji wszystkich agentów.

| | |
|---|---|
| $\mathsf{calm} \xrightarrow{(\texttt{detectIrritation},*)} \mathsf{irritated}$ | $\mathsf{calm} \xrightarrow{(\texttt{noop},*)} \mathsf{calm}$ |
| $\mathsf{irritated} \xrightarrow{(\texttt{detectCalm},*)} \mathsf{calm}$ | $\mathsf{irritated} \xrightarrow{(\texttt{noop},*)} \mathsf{irritated}$ |
| $\mathsf{calm} \xrightarrow{(\texttt{detectBoredom},*)} \mathsf{bored}$ | $\mathsf{bored} \xrightarrow{(\texttt{noop},*)} \mathsf{bored}$ |
| $\mathsf{bored} \xrightarrow{(\texttt{detectCalm},*)} \mathsf{calm}$ | |

Tabela 2.1: Tranzycje agenta Auto.

### Agent Radio

Zakładamy, że agent Radio może znaleźć się w jednym z trzech stanów emocjonalnych takich samych jak agent Auto: $\{\mathsf{irritated}, \mathsf{calm}, \mathsf{bored}\}$ oraz w jednym z trzech stanów wyrażających jaka muzyka jest aktualnie grana: $\{\mathsf{relaxing}, \mathsf{pop}, \mathsf{dynamic}\}$. Są to stany, w których prawdziwe są zmienne o tych samych nazwach. Razem daje to dziewięć stanów lokalnych:

$$ELoc_{Radio} = \{(\mathsf{relaxing}, \mathsf{irritated}), (\mathsf{relaxing}, \mathsf{calm}), (\mathsf{relaxing}, \mathsf{bored}), (\mathsf{pop}, \mathsf{irritated}),$$

$$(\mathsf{pop}, \mathsf{calm}), (\mathsf{pop}, \mathsf{bored}), (\mathsf{dynamic}, \mathsf{irritated}), (\mathsf{dynamic}, \mathsf{calm}), (\mathsf{dynamic}, \mathsf{bored})\}.$$

Agent zaczyna w stanie początkowym: $(\mathsf{pop}, \mathsf{calm})$.
Dostępne są trzy akcje wskazujące na graną muzykę oraz jedna akcja oznaczająca brak nowych operacji:

$$Act_{Radio} = \{\texttt{playRelaxing}, \texttt{playPop}, \texttt{playDynamic}, \texttt{noop}\}.$$

Możliwe przejścia między stanami i równocześnie protokół zadane są tranzycjami opisanymi w tabeli 2.2, gdzie $\delta \in \{\texttt{relaxing}, \texttt{pop}, \texttt{dynamic}\}$ oraz $\upsilon \in \{\texttt{irritated}, \texttt{calm}, \texttt{bored}\}$.

$$(\delta, \texttt{irritated}) \xrightarrow{(\texttt{noop},\texttt{playRelaxing})} (\texttt{relaxing}, \texttt{irritated})$$
$$(\delta, \texttt{calm}) \xrightarrow{(\texttt{noop},\texttt{playPop})} (\texttt{pop}, \texttt{calm})$$
$$(\delta, \texttt{bored}) \xrightarrow{(\texttt{noop},\texttt{playDynamic})} (\texttt{dynamic}, \texttt{bored})$$
$$(\delta, \upsilon) \xrightarrow{(\texttt{detectIrritated},\texttt{noop})} (\delta, \texttt{irritated})$$
$$(\delta, \upsilon) \xrightarrow{(\texttt{detectCalm},\texttt{noop})} (\delta, \texttt{calm})$$
$$(\delta, \upsilon) \xrightarrow{(\texttt{detectBored},\texttt{noop})} (\delta, \texttt{bored})$$

Tabela 2.2: Tranzycje agenta Radio.

Zastosowanie akcji `noop` wywołało pewien rodzaj asynchroniczności w systemie, powodując naprzemienność wykonywanych akcji. Można tego uniknąć, pozwalając na jednoczesne wykonanie akcji typu `detect`$\star$ oraz `play`$\star$. System wieloagentowy, składający się z opisanych agentów, funkcjonuje zgodnie z definicjami przedstawionymi w rozdziale 2.2. Na tych samych podstawach formalnych oparta jest również konstrukcja modelu systemu.

## 2.8. Przegląd literatury

Obecnie, w takim rozumieniu, w jakim funkcjonują dojrzałe narzędzia dla logik takich jak ATL czy CTL, nie istnieje jedno powszechnie akceptowane, kompleksowe narzędzie umożliwiające testowanie spełnialności oraz przeprowadzanie syntezy dla logik wieloagentowych uwzględniających emocje. Dostępne rozwiązania skupiają się przede wszystkim na symulacji emocjonalnych zachowań agentów, a nie na formalnym wnioskowaniu logicznym. Brakuje narzędzi implementujących rozszerzenia logik formalnych (np. ATL czy BDI) o komponent emocjonalny, które umożliwia-

łyby automatyczne sprawdzanie spełnialności formuł bądź syntezę strategii. Z tego względu niniejszy przegląd literatury obejmuje z jednej strony istniejące formalne modele emocji wykorzystywane również w kontekście systemów wieloagentowych, a z drugiej analizę problemu spełnialności w logice ATL oraz narzędzi służących do jej testowania i przeprowadzania syntezy modeli.

### 2.8.1. Modele emocji

Współczesne podejścia do formalnego modelowania emocji w systemach wieloagentowych bardzo często opierają się na teorii oceny emocji (ang. *appraisal theory*) Ortony, Clore i Collins (1998); Scherer, Schorr i Johnstone (2001). Zgodnie z założeniem tego nurtu, kluczowym czynnikiem wpływającym na intensywność emocji są przekonania i intencje agentów. Emocje nie powstają w oderwaniu od kontekstu, lecz są reakcją na konkretne zdarzenia oraz ich konsekwencje – przede wszystkim w odniesieniu do możliwości realizacji zamierzonych celów Castelfranchi (2000); De Rosis, Pelachaud, Poggi, Carofiglio i De Carolis (2003). W rezultacie w systemach formalnych emocje są często definiowane poprzez stany mentalne agentów, takie jak przekonania, pragnienia i intencje.

Ważnym nurtem w tej dziedzinie jest modelowanie emocji w ramach architektury BDI (ang. *beliefs–desires–intentions*), łączące zarówno podstawy empiryczne, jak i teoretyczne. Przykładem takiego podejścia jest praca Ochs, Sadek i Pelachaud (2012), w której zaproponowano formalną, semantycznie ugruntowaną reprezentację racjonalnych agentów dialogowych. Model ten pozwala agentom nie tylko na rozpoznawanie kontekstu emocjonalnego sytuacji, lecz także na wyrażanie empatii, gdy jest to adekwatne. Architektura BDI została także zastosowana w systemach edukacyjnych, m.in. w pracy Jaques i Viccari (2004), gdzie posłużyła do diagnozy emocjonalnej ucznia w interakcji z wirtualnym agentem pedagogicznym.

Kolejne rozszerzenie koncepcji emocjonalnych agentów przedstawiono w pracy Meyer (2004), gdzie zaadaptowano teorię komunikacyjną emocji Oatleya Oatley (1992). W tym modelu emocje są wyrażane poprzez postawy mentalne zapisane w logice modalnej, a ich zadaniem jest kierowanie działaniami agenta w określonym kontekście sytuacyjnym. Duże znaczenie w literaturze przypisuje się także modelowi OCC autorstwa Ortony'ego, Clore'a i Collinsa Ortony i in. (1998), który znalazł zastosowanie m.in. w pracy Adam, Gaudou, Herzig i Longin (2006). Model ten wyróżnia 22 kategorie emocji oraz pięć procesów odpowiedzialnych za ich generowanie i regulację. Emocje w ujęciu OCC zależą nie tylko od samych zdarzeń, lecz również od obecności innych agentów i obiektów w środowisku, a także od relacji między nimi.

Aktualnie coraz większą uwagę poświęca się systemom wyposażonym w agentów zdolnych do emocjonalnej interakcji. Obecność emocji w zachowaniu agentów

znacząco zwiększa ich ekspresyjność, adaptacyjność i wiarygodność – cechy kluczowe dla tworzenia bardziej naturalnych interakcji człowiek–maszyna. Szczególne znaczenie ma to w kontekście projektowania wirtualnych agentów ucieleśnionych, których zadaniem jest budowanie trwałej relacji z użytkownikiem. Ponadto modele emocji znajdują zastosowanie w testowaniu hipotez z zakresu psychologii emocji, rozwijaniu immersyjnych doświadczeń w grach komputerowych oraz w doskonaleniu algorytmów podejmowania decyzji Becker, Kopp i Wachsmuth (2004); El-Nasr, Yen i Ioerger (2000); Nawwab, Dunne i Bench-Capon (2010); Silveira, da Silva Bitencourt, Gelaim, Marchi i de la Prieta (2016).

Najnowsze podejścia do formalizacji emocji koncentrują się na integracji mechanizmów emocjonalnych z racjonalnym podejmowaniem decyzji przez agentów. Coraz częściej stosuje się architektury BDI rozszerzone o komponenty afektywne Ochs i in. (2012), jak również modele samouczące się, które odwzorowują ludzkie reakcje emocjonalne na podstawie danych i interakcji Hernández-Marcos i Ros (2024). Dużą rolę odgrywają także systemy wieloagentowe, w których emocje wspierają procesy komunikacji, negocjacji i adaptacji społecznej Ortigoso i in. (2025); Perez, Argente, Del Val Noguera i Valero (2022). Celem tych badań jest nie tylko poprawa naturalności interakcji człowiek–maszyna, lecz także lepsze odwzorowanie złożonych zjawisk psychologicznych w środowiskach symulacyjnych.

Praca Bustos i in. (2025) wnosi istotny wkład w modelowanie emocji w kontekście środowisk dynamicznych, pokazując, w jaki sposób czynniki zewnętrzne, takie jak sytuacja na drodze, wpływają na subiektywny poziom stresu kierowców. Wyniki badań potwierdzają, że emocje mogą być skutecznie modelowane i przewidywane na podstawie danych kontekstowych, co stanowi ważny krok w kierunku tworzenia systemów afektywnych zdolnych do adaptacji do warunków otoczenia.

Nowe trendy skupiają się także wokół modelowania opartego na wieloagentowej logice probabilistycznej afektywnej (ang. *Affective Probabilistic Logic*) Wang, Liu, Liu, Samsonovich i Klimov (2024). Element probabilistyczny umożliwia modelowanie niepewności towarzyszącej emocjom i ich wpływowi na decyzje agentów. Dzięki temu możliwe jest bardziej realistyczne odwzorowanie dynamiki stanów emocjonalnych, uwzględniające stopień prawdopodobieństwa wystąpienia określonych emocji oraz ich zróżnicowany wpływ na zachowanie agentów w systemie.

Innym ważnym elementem współczesnych badań jest modelowanie emocji w konwersacjach, pokazujące jak systemy oparte na sztucznej inteligencji mogą wspierać rozwój samoświadomości emocjonalnej poprzez dialog Shen, Lecamwasam, Park, Breazeal i Picard (2023). Przeprowadzone doświadczenia wskazują, że personalizacja i responsywność emocjonalna agenta konwersacyjnego odgrywają kluczową rolę w skutecznym wspieraniu użytkowników w rozpoznawaniu i werbalizowaniu emocji.

### 2.8.2. Spełnialność i synteza logiki ATL

Problem syntezy stanowi jedno z kluczowych zagadnień w dynamicznie rozwijającej się dziedzinie sztucznej inteligencji oraz współczesnej inżynierii oprogramowania Jones, Knapik, Penczek i Lomuscio (2012); Kouvaros, Lomuscio i Pirovano (2018); Schewe i Finkbeiner (2007). Celem tych badań jest automatyczne tworzenie nowatorskiego oprogramowania, również dla agentów AI, chatbotów czy autonomicznych pojazdów. Wygodnym formalizmem do specyfikacji interakcji w systemach rozproszonych, mających charakter gry, jest logika czasowa ATL Alur i in. (1997). Logika ta została zaprojektowana do rozumowania o zdolnościach strategicznych agentów i ich koalicji. Powstało wiele prac analizujących różne wersje ATL Belardinelli, Lomuscio, Murano i Rubin (2019); Bulling, Dix i Jamroga (2010); Dima, Maubert i Pinchinat (2014); Dima i Tiplea (2011); Jamroga, Knapik i Kurpiewski (2017); Schobbens (2004), a także inne logiki modalne opisujące zdolności strategiczne Chatterjee, Henzinger i Piterman (2010); Mogavero, Murano, Perelli i Vardi (2012a, 2012b). Ze względu na złożoność obliczeniową tych logik, wciąż istnieje potrzeba rozwijania nowych technik rozwiązywania problemów spełnialności i syntezy Bloem, Jobstmann, Piterman, Pnueli i Sa'ar (2012); Bloem, Könighofer i Seidl (2014); Finkbeiner i Schewe (2013); Kupferman i Vardi (2005) i poszukiwania wydajnych algorytmów praktycznych. Jednym z ważniejszych podejść jest zastosowanie techniki SMT (ang. *Satisfiability Modulo Theories*) oraz algorytmów parametrycznej ograniczonej weryfikacji modelowej (ang. *Parametric Bounded Model Checking*), które oferują efektywną metodę syntezy dla egzystencjalnego fragmentu logiki SMTL (ang. *Strategic Metric Temporal Logic*) Kacprzak, Niewiadomski, Penczek i Zbrzezny (2023); Niewiadomski i in. (2024).

Złożoność problemu spełnialności dla logiki ATL z pełną informacją i bez pamięci, jest bardzo wysoka - wykazano, że problem jest EXPTIME-zupełny (ang. *EXPTIME-complete*) dla stałej liczby agentów Goranko i Drimmelen (2006); van Drimmelen (2003), a później rozszerzono ten wynik na przypadek ogólny Walther, Lutz, Wolter i Wooldridge (2006). Natomiast dla logiki ATL$^*$ z pełną informacją udowodniono zupełność względem klasy 2EXPTIME Schewe (2008). W dowodach tych wykorzystywano techniki oparte na automatach drzew alternujących. Konstruktywna metoda decyzyjna oparta na tablicach została zaproponowana przez Goranko i Shkatov (2009), a następnie rozszerzona na ATL$^*$ David (2015) oraz ATEL - epistemiczne rozszerzenie ATL Belardinelli (2014). Złożoność (a nawet rozstrzygalność) problemu spełnialności dla ATL z niepełną informacją, pozostaje nieznana. Przypuszczalną trudność ilustrują wyniki Jamroga i Dix (2006); Schobbens (2004), które pokazują, że problem weryfikacji modelowej dla ATL z niepełną informacją jest zupełny względem klasy $\Delta_2^P$, a sama logika nie posiada standardowej charakterystyki punktu stałego Bulling i Jamroga (2011); Dima, Maubert i Pinchinat (2015).

W pracach Kacprzak i in. (2020); Niewiadomski i in. (2020) zaprezentowano narzędzie dedykowane obu wariantom ATL, z pełną i niepełną informacją, oparte na

metodzie SAT Modulo Monotonic Theories (SMMT) Klenze i in. (2016), stosowanej wcześniej do poszukiwania modeli formuł logiki CTL. Technika ta, wprowadzona przez Baylessa i in. Bayless i in. (2015), została rozwinięta przez Klenzego i in. Klenze i in. (2016), którzy stworzyli wydajny solver SMT dla teorii weryfikacji modeli w CTL i zastosowali go również do efektywnej syntezy w tej logice.

Innym narzędziem jest TATL David (2015), implementujące procedurę decyzyjną opartą na metodzie tablicowej, i jest pierwszym narzędziem służącym do badania spełnialności formuł logiki ATL$^*$. Formuły interpretowane są w modelach gier współbieżnych (ang. *Concurrent Game Models*) z pełną pamięcią i informacją. Główna idea tego podejścia polega na budowie ukorzenionego grafu skierowanego na podstawie formuły $\phi$, z którego można wydobyć model gry spełniający $\phi$.

Trzecim narzędziem służącym do badania spełnialności formuł jest *Coalgebraic Ontology Logic Reasoner* (COOL) Hausmann, Schröder i Egger (2016). Narzędzie to implementuje wydajną procedurę decyzyjną dla spełnialności formuł w *alternation-free* $\mu$*-calculus*, opartą na technice globalnego buforowania. Metoda ta pozwala efektywnie unikać powtórnych obliczeń podczas konstruowania modelu, co znacząco poprawia skalowalność procesu rozumowania. Należy jednak zaznaczyć, że COOL obsługuje wyłącznie logikę koalicji Pauly (2002), która jest bezczasowym fragmentem ATL, a zatem nie pozwala na wnioskowanie o dynamice systemu.

## Podsumowanie

W ostatnich dekadach logiki formalne stały się podstawowym narzędziem opisu i analizy złożonych systemów rozproszonych, systemów reaktywnych oraz środowisk wieloagentowych. W szczególności logiki temporalne, takie jak LTL czy CTL, są szeroko stosowane w weryfikacji modelowej i automatycznej analizie systemów. Rozszerzeniem tych logik na potrzeby systemów wieloagentowych są logiki strategiczne, takie jak ATL. Umożliwiają one analizę możliwości koalicji agentów w osiąganiu celów przy założeniu różnego typu strategii oraz informacji.

Istotnym aspektem obecnych badań jest także modelowanie emocji i stanów afektywnych w systemach wieloagentowych. Choć klasyczne podejścia logiczne nie uwzględniają komponentów emocjonalnych, w ostatnich latach pojawiły się propozycje rozszerzeń logik modalnych o elementy afektywne. Prace te często opierają się na integracji formalnych modeli logiki z psychologicznymi teoriami emocji oraz systemami symulacyjnymi.

Niniejsza praca sytuowana jest na styku wspomnianych kierunków: łączy klasyczne podejście do syntezy i spełnialności w logikach modalnych ze współczesnymi wymaganiami dotyczącymi modelowania agentów afektywnych. Proponowana metoda syntezy modeli afektywnych agentów w logice ATL stanowi nowe ujęcie problemu i może być rozwijana w kierunku zastosowań praktycznych, takich jak projektowanie

systemów interakcji człowiek-komputer, robotyka społeczna czy symulacja procesów decyzyjnych.

Wkład pracy polega na rozszerzeniu metody syntezy modeli opracowanej dla logiki ATL o komponent afektywny, umożliwiający modelowanie systemów wieloagentowych, w których decyzje agentów zależą nie tylko od strategii racjonalnych, lecz również od ich stanów emocjonalnych. Zaproponowane podejście obejmuje formalne rozszerzenie klasycznego modelu MAS o zmienne reprezentujące natężenie emocji, przy jednoczesnym zachowaniu spójności semantycznej logiki ATL. Uogólnienie metody wymagało dostosowania przestrzeni stanów lokalnych oraz redefinicji funkcji przejścia i protokołu tak, aby akcje mogły wpływać na komponent afektywny. Kluczowym rezultatem jest wykazanie, że rozszerzenie to nie narusza własności monotoniczności, dzięki czemu technika SMMT pozostaje w pełni stosowalna. Proponowana metoda stanowi zatem spójne rozwinięcie klasycznej koncepcji ATL, zachowujące poprawność formalną i umożliwiające analizę systemów kognitywnych. W rezultacie zaproponowane rozwiązanie stanowi rozwinięcie metody opracowanej dla logiki ATL Kacprzak i in. (2020), Niewiadomski i in. (2020) oraz jej uogólnienia SL Kacprzak i in. (2021), wzbogacające ją o możliwość modelowania aspektów emocjonalnych przy zachowaniu pełnej zgodności z dotychczasowym aparatem formalnym i narzędziowym.

Praca przedstawia teoretyczne podstawy oraz opis metody, w związku z czym stanowi fundament pod dalsze badania w zakresie formalizacji emocji w systemach wieloagentowych. Kolejnym krokiem w jej rozwoju będzie implementacja modułu syntezy modeli dla afektywnych systemów agentowych umożliwiającego automatyczne generowanie struktur spełniających zadane własności logiczne, przy jednoczesnym uwzględnieniu komponentów emocjonalnych. Rozbudowa ta stanowi naturalne rozszerzenie zaprezentowanego podejścia i otwiera drogę do praktycznego zastosowania formalnych logik emocji w inteligentnych systemach interaktywnych.

## Podziękowania

Badania zostały zrealizowane w ramach pracy nr WZ/WI-IIT/2/2025 w Politechnice Białostockiej i sfinansowane z subwencji badawczej przekazanej przez ministra właściwego do spraw nauki.

## Bibliografia

Adam, C., Gaudou, B., Herzig, A. i Longin, D. (2006). OCC's emotions: a formalization in a BDI logic. W: *International conference on artificial intelligence:*

*Methodology, systems, and applications* 24–32.

Affectiva2025. (2025). *Affectiva automotive AI: Emotion AI for driver monitoring systems.* https://www.affectiva.com/product/affectiva-automotive-ai-for-driver-monitoring-solutions/. (Dostęp online: July 2025)

Alur, R., Henzinger, T. A. i Kupferman, O. (1997). Alternating-time Temporal Logic. W: *Proceedings of the 38th annual symposium on foundations of computer science (focs)* 100–109.

Alur, R., Henzinger, T. A. i Kupferman, O. (2002). Alternating-time Temporal Logic. *Journal of the ACM*, *49*(5), 672–713.

Baier, C., i Katoen, J.-P. (2008). *Principles of model checking*. MIT Press.

Bayless, S., Bayless, N., Hoos, H. i Hu, A. (2015). SAT modulo monotonic theories. W: *Proceedings of the twenty-ninth aaai conference on artificial intelligence* 3702–3709.

Becker, C., Kopp, S. i Wachsmuth, I. (2004). Simulating the emotion dynamics of a multimodal conversational agent. W: *Workshop on affective dialogue systems* 154–165.

Belardinelli, F. (2014). Reasoning about knowledge and strategies: Epistemic strategy logic. W: *Proceedings 2nd international workshop on strategic reasoning, SR 2014, grenoble, france, april 5-6, 2014* 27–33.

Belardinelli, F., Lomuscio, A., Murano, A. i Rubin, S. (2019). Imperfect information in alternating-time temporal logic on finite traces. W: *PRIMA 2019: Principles and practice of multi-agent systems - 22nd international conference, turin, italy, october 28-31, 2019, proceedings* 469–477.

Biere, A., Heule, M., van Maaren, H. i Walsh, T. (2021). *Handbook of satisfiability*. IOS Press.

Biere, A., Järvisalo, M. i Kiesl, B. (2021). Preprocessing in sat solving. W: *Handbook of satisfiability* 391–435.

Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A. i Saar, Y. (2012). Synthesis of reactive(1) designs. W: *Formal methods in computer aided design (fmcad)* 15–22.

Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A. i Sa'ar, Y. (2012). Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, *78*, 911-938.

Bloem, R., Könighofer, R. i Seidl, M. (2014). SAT-based synthesis methods for safety specs. W: *International conference on verification, model checking, and abstract interpretation* 1–20.

Brave, S., i Nass, C. (2002). Emotion in human-computer interaction. W: *The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications* 81–96.

Bulling, N., Dix, J. i Jamroga, W. (2010). Model checking logics of strategic ability: Complexity. W: M. Dastani, K. Hindriks i J.-J. Meyer (red.), *Specification and verification of multi-agent systems* 125–159.

Bulling, N., i Jamroga, W. (2011). Alternating epistemic mu-calculus. W: *Proce-*

*edings of IJCAI-11* 109–114.

Bustos, C., Solé-Ribalta, A., Haouij, N. E., Borge-Holthoefer, J., Lapedriza, À. i Picard, R. W. (2025). Analyzing the visual road scene for driver stress estimation. *IEEE Trans. Affect. Comput.*, *16*(3), 1787–1801.

Castelfranchi, C. (2000). Affective appraisal versus cognitive evaluation in social emotions and interactions. W: *Affective interactions: Towards a new generation of computer interfaces* 76–106.

Chatterjee, K., Henzinger, T. i Piterman, N. (2010). Strategy logic. *Inf. Comput.*, *208*(6), 677–693.

Clarke, E., i Emerson, E. (1981). Design and synthesis of synchronization skeletons using branching time temporal logic. W: *Proceedings of logics of programs workshop* 131, 52–71.

Clarke, E. M., Grumberg, O., Kroening, D., Peled, D. A. i Veith, H. (2018). *Model checking, 2nd edition*. MIT Press.

Cook, S. A. (1971). The complexity of theorem-proving procedures. *STOC*, 151–158.

David, A. (2015). Deciding ATL* satisfiability by tableaux. W: *International conference on automated deduction* 214–228.

De Rosis, F., Pelachaud, C., Poggi, I., Carofiglio, V. i De Carolis, B. (2003). From Greta's mind to her face: modelling the dynamics of affective states in a conversational embodied agent. *International journal of human-computer studies*, *59*(1), 81–118.

Dima, C., Maubert, B. i Pinchinat, S. (2014). The expressive power of epistemic $\mu$-calculus. *CoRR*, *abs/1407.5166*.

Dima, C., Maubert, B. i Pinchinat, S. (2015). Relating paths in transition systems: The fall of the modal mu-calculus. W: *Proceedings of MFCS* 9234, 179–191. doi: 10.1007/978-3-662-48057-1_14

Dima, C., i Tiplea, F. (2011). Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR*, *abs/1102.4225*.

Ekman, P. (1992). An argument for basic emotions. *Cognition and emotion*, *6*, 169-200.

El-Nasr, M. S., Yen, J. i Ioerger, T. R. (2000). Flame—fuzzy logic adaptive model of emotions. *Autonomous Agents and Multi-Agent Systems*, *3*(3), 219–257.

Enderton, H. B. (2001). *A mathematical introduction to logic*. Academic Press.

Eye, S. (2025). *Interior sensing for automotive.* `https://www.smarteye.se/solutions/automotive/interior-sensing/`. (Dostęp online: July 2025)

Finkbeiner, B., i Schewe, S. (2013). Bounded synthesis. *International Journal on Software Tools for Technology Transfer*, *15*(5-6), 519–539.

Ghallab, M., Nau, D. i Traverso, P. (2004). Automated planning: theory and practice. *Elsevier*.

Goranko, V., i Drimmelen, G. V. (2006). Complete axiomatization and decidability of alternating-time temporal logic. *Theoretical Computer Science*, *353*(1-3),

93–117.

Goranko, V., i Shkatov, D. (2009). Tableau-based decision procedures for logics of strategic ability in multiagent systems. *ACM Trans. Comput. Log.*, *11*(1), 3:1–3:51.

Hausmann, D., Schröder, L. i Egger, C. (2016). Global caching for the alternation-free $\mu$-calculus. W: J. Desharnais i R. Jagadeesan (red.), *27th international conference on concurrency theory, CONCUR 2016, august 23-26, 2016, québec city, canada* 59, 34:1–34:15.

Healey, J. A., i Picard, R. W. (2005). Detecting stress during real-world driving tasks using physiological sensors. *IEEE Transactions on Intelligent Transportation Systems*, *6*(2), 156–166. doi: 10.1109/TITS.2005.848368

Hernández-Marcos, A., i Ros, E. (2024). A generic self-learning emotional framework for machines. *Scientific Reports*, *14*(1), 1–13.

Jamroga, W., i Dix, J. (2006). Model checking $\mathrm{ATL}_{ir}$ is indeed $\Delta_2^P$-complete. W: *Proceedings of eumas'06* 223.

Jamroga, W., Knapik, M. i Kurpiewski, D. (2017). Fixpoint approximation of strategic abilities under imperfect information. W: *Proceedings of the 16th conference on autonomous agents and multiagent systems, AAMAS 2017, são paulo, brazil, may 8-12, 2017* 1241–1249.

Jamroga, W., Knapik, M., Kurpiewski, D. i Mikulski, Ł. (2019). Approximate verification of strategic abilities under imperfect information. *Artif. Intell.*, *277*.

Jaques, P. A., i Viccari, R. M. (2004). A BDI approach to infer student's emotions. W: *Ibero-american conference on artificial intelligence* 901–911.

Jones, A. V., Knapik, M., Penczek, W. i Lomuscio, A. (2012). Group synthesis for parametric temporal-epistemic logic. W: *International conference on autonomous agents and multiagent systems, AAMAS 2012, valencia, spain, june 4-8, 2012 (3 volumes)* 1107–1114.

Kacprzak, M. (2017). Persuasive strategies in dialogue games with emotional reasoning. W: L. Polkowski i in. (red.), *Rough sets - international joint conference, IJCRS 2017, olsztyn, poland, july 3-7, 2017, proceedings, part II* 10314, 435–453.

Kacprzak, M., Niewiadomski, A. i Penczek, W. (2020). SAT-Based ATL Satisfiability Checking. W: *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning* 539–549.

Kacprzak, M., Niewiadomski, A. i Penczek, W. (2021). Satisfiability checking of strategy logic with simple goals. W: M. Bienvenu, G. Lakemeyer i E. Erdem (red.), *Proceedings of the 18th international conference on principles of knowledge representation and reasoning, KR 2021, online event, november 3-12, 2021* 400–410.

Kacprzak, M., Niewiadomski, A., Penczek, W. i Zbrzezny, A. (2023). SMT-based satisfiability checking of Strategic Metric Temporal Logic. W: K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein i R. Radulescu (red.), *ECAI 2023 - 26th european*

*conference on artificial intelligence, september 30 - october 4, 2023, kraków, poland* 372, 1180–1189.

Klenze, T., Bayless, S. i Hu, A. (2016). Fast, flexible, and minimal CTL synthesis via SMT. W: S. Chaudhuri i A. Farzan (red.), *Computer aided verification* 136–156.

Kouvaros, P., Lomuscio, A. i Pirovano, E. (2018). Symbolic synthesis of fault-tolerance ratios in parameterised multi-agent systems. W: *Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI 2018, july 13-19, 2018, stockholm, sweden* 324–330.

Kupferman, O., i Vardi, M. (2005). Safraless decision procedures. W: *46th annual ieee symposium on foundations of computer science (focs'05)* 531–540.

Kurpiewski, D., Jamroga, W. i Knapik, M. (2019). STV: model checking for strategies under imperfect information. W: *Proceedings of the 18th international conference on autonomous agents and multiagent systems, AAMAS '19, montreal, qc, canada, may 13-17, 2019* 2372–2374.

Lomuscio, A., Qu, H. i Raimondi, F. (2017). MCMAS: an open-source model checker for the verification of multi-agent systems. *International Journal on Software Tools for Technology Transfer*, *19*(1), 9–30.

Meyer, J.-J. C. (2004). Reasoning about emotional agents. W: *Proceedings of the 16th european conference on artificial intelligence* 129–133.

Mogavero, F., Murano, A., Perelli, G. i Vardi, M. (2012a). A decidable fragment of strategy logic. *CoRR*, *abs/1202.1309*.

Mogavero, F., Murano, A., Perelli, G. i Vardi, M. Y. (2012b). What makes ATL* decidable? a decidable fragment of strategy logic. W: *Proceedings of concur* 193–208.

Mogavero, F., Murano, A., Perelli, G. i Vardi, M. Y. (2014a). Reasoning about strategies: On the model-checking problem. *ACM Trans. Comput. Log.*, *15*(4), 34:1–34:47.

Mogavero, F., Murano, A., Perelli, G. i Vardi, M. Y. (2014b). Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic (TOCL)*, *15*(4), 1–47.

Nawwab, F. S., Dunne, P. E. i Bench-Capon, T. (2010). Exploring the role of emotions in rational decision making. W: *Proc. of comma* 367–378.

Niewiadomski, A., Kacprzak, M., Kurpiewski, D., Knapik, M., Penczek, W. i Jamroga, W. (2020). MsATL: A tool for SAT-based ATL satisfiability checking. W: A. E. F. Seghrouchni, G. Sukthankar, B. An i N. Yorke-Smith (red.), *Proceedings of the 19th international conference on autonomous agents and multiagent systems, AAMAS* 2111–2113.

Niewiadomski, A., Nazarczuk, M., Przychodzki, M., Kacprzak, M., Penczek, W. i Zbrzezny, A. (2024). SMT4SMTL: A tool for SMT-based satisfiability checking of SMTL. W: M. Dastani, J. S. Sichman, N. Alechina i V. Dignum (red.), *Proceedings of the 23rd international conference on autonomous agents*

*and multiagent systems, AAMAS 2024, auckland, new zealand, may 6-10, 2024* 2815–2817.

Oatley, K. (1992). *Best laid schemes: The psychology of the emotions*. Cambridge University Press.

Ochs, M., Sadek, D. i Pelachaud, C. (2012). A formal model of emotions for an empathic rational dialog agent. *Autonomous Agents and Multi-Agent Systems*, *24*(3), 410–440.

Ortigoso, A. R., Vieira, G., Fuentes, D., Frazão, L., Costa, N. i Pereira, A. (2025). Project riley: Multimodal multi-agent llm collaboration with emotional reasoning and voting. *arXiv preprint arXiv:2505.20521*.

Ortony, A., Clore, G. L. i Collins, A. (1998). *The cognitive structure of emotions*. Cambridge University Press, United Kingdom.

Pauly, M. (2002). A modal logic for coalitional power in games. *J. Log. Comput.*, *12*(1), 149–166.

Perez, D., Argente, E., Del Val Noguera, E. i Valero, S. (2022). *Introducing emotions in the reasoning cycle of normative aware agents.* `https://arxiv.org/abs/2209.10049`.

Picard, R. W. (1997). Affective computing. *MIT Press*.

Pnueli, A. (1977). The temporal logic of programs. W: *18th annual symposium on foundations of computer science (focs)* 46–57. doi: 10.1109/SFCS.1977.32

Pnueli, A., i Rosner, R. (1989). On the synthesis of a reactive module. *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 179–190.

Scherer, K. R., Schorr, A. i Johnstone, T. (2001). *Appraisal processes in emotion: Theory, methods, research*. Oxford University Press.

Schewe, S. (2008). ATL* satisfiability is 2EXPTIME-complete. W: *Automata, languages and programming, 35th international colloquium, ICALP 2008, reykjavik, iceland, july 7-11, 2008, proceedings, part II - track B: logic, semantics, and theory of programming & track C: security and cryptography foundations* 5126, 373–385.

Schewe, S., i Finkbeiner, B. (2007). Distributed synthesis for alternating-time logics. W: *Automated technology for verification and analysis, 5th international symposium, ATVA 2007, tokyo, japan, october 22-25, 2007, proceedings* 268–283.

Schobbens, P. Y. (2004). Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, *85*(2), 82–93.

Shen, J., Lecamwasam, K., Park, H. W., Breazeal, C. i Picard, R. W. (2023). Designing conversational agents for emotional self-awareness. W: *11th international conference on affective computing and intelligent interaction, ACII 2023 - workshops and demos, cambridge, ma, usa, september 10-13, 2023* 1–4.

Silveira, R., da Silva Bitencourt, G. K., Gelaim, T. A., Marchi, J. i de la Prieta, F. (2016). Towards a model of open and reliable cognitive multiagent systems:

Dealing with trust and emotions. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *4*(3).

Tabuada, P. (2009). *Verification and control of hybrid systems: A symbolic approach*. Springer.

Tao, J., i Tan, T. (2005). Affective computing: A review. W: J. Tao, T. Tan i R. W. Picard (red.), *Affective computing and intelligent interaction* 981–995.

van Drimmelen, G. (2003). Satisfiability in alternating-time temporal logic. W: *18th annual ieee symposium of logic in computer science, 2003. proceedings.* 208–217.

Walther, D., Lutz, C., Wolter, F. i Wooldridge, M. (2006). ATL satisfiability is indeed EXPTIME-complete. *Journal of Logic and Computation*, *16*(6), 765-787.

Wang, Y., Liu, Z., Liu, T., Samsonovich, A. V. i Klimov, V. V. (2024). On the logic of agent's emotions. *Cognitive Systems Research*, *88*, 101281.

Zepf, S., Hernandez, J., Schmitt, A., Minker, W. i Picard, R. W. (2020). Driver emotion recognition for intelligent vehicles: A survey. , *53*(3).