IEEE World Congress on Computational Intelligence
24-29 July 2016, Vancouver, Canada
IEEE WCCI 2016
Vancouver Canada

FUZZ-IEEE 2016 Tutorial: FUZZ-IEEE-03

Type-2 Fuzzy Ontology and Fuzzy Markup Language for Real-World Applications

Organized by

Chang-Shing Lee, National University of Tainan, Taiwan
Giovanni Acampora, Nottingham Trent University, UK
Yuandong Tian, Facebook AI Research, USA

24 July, 2016

IEEE World Congress on Computational Intelligence
24-29 July 2016, Vancouver, Canada
IEEE WCCI 2016
Vancouver Canada

FUZZ-IEEE 2016 Tutorial: FUZZ-IEEE-03

Part 1: Type-2 Fuzzy Ontology and Applications
Chang-Shing Lee, NUTN, Taiwan

Part 2: Fuzzy Markup Language
Giovanni Acampora, NTU, UK

Part 3: Real-World Application on Game of Go
Yuandong Tian, Facebook AI Research, USA

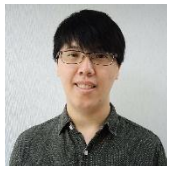
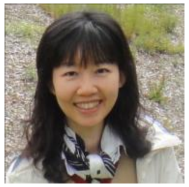
IEEE World Congress on Computational Intelligence
24-29 July 2016, Vancouver, Canada
IEEE WCCI 2016
Vancouver Canada

FUZZ-IEEE 2016 Tutorial: FUZZ-IEEE-03 Part 1

Type-2 Fuzzy Ontology and Applications

Chang-Shing Lee
National University of Tainan, Taiwan

Research Team

Co-Sponsors

IEEE Computational Intelligence Society	IEEE	科技部 Ministry of Science and Technology	
人工知能学会 JSAI	K.W.S Center for Research of Knowledge Application and Web Service 知識應用暨網路服務研究中心	高雄市政府教育局 EDUCATION BUREAU, KAOHSIUNG CITY GOVERNMENT	NARLabs 國家實驗研究院 國家高速網路與計算中心

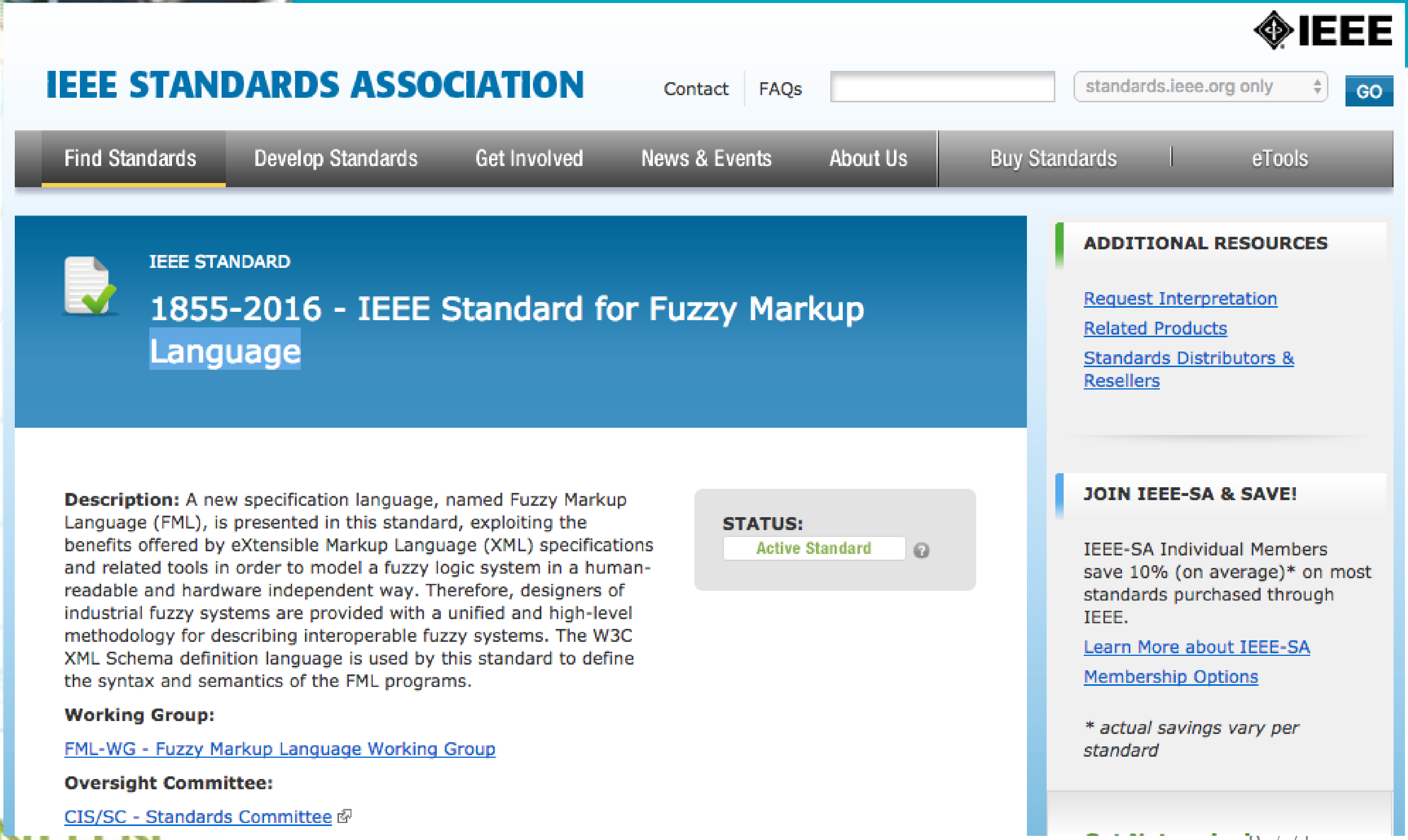
K.W.S

NUTN

Type-2 Fuzzy Ontology Applications

- **FML IEEE 1855-2016 Standard**
- **Type-2 Fuzzy Set**
- **Fuzzy Ontology**
- **Game of Go Application**
- **Personalized Diet Recommendation**
- **Adaptive Learning Application**

FML IEEE 1855-2016



IEEE STANDARD

1855-2016 - IEEE Standard for Fuzzy Markup Language

Description: A new specification language, named Fuzzy Markup Language (FML), is presented in this standard, exploiting the benefits offered by eXtensible Markup Language (XML) specifications and related tools in order to model a fuzzy logic system in a human-readable and hardware independent way. Therefore, designers of industrial fuzzy systems are provided with a unified and high-level methodology for describing interoperable fuzzy systems. The W3C XML Schema definition language is used by this standard to define the syntax and semantics of the FML programs.

Working Group:

FML-WG - Fuzzy Markup Language Working Group

Oversight Committee:

CIS/SC - Standards Committee

STATUS:
Active Standard

ADDITIONAL RESOURCES

Request Interpretation
Related Products
Standards Distributors & Resellers

JOIN IEEE-SA & SAVE!

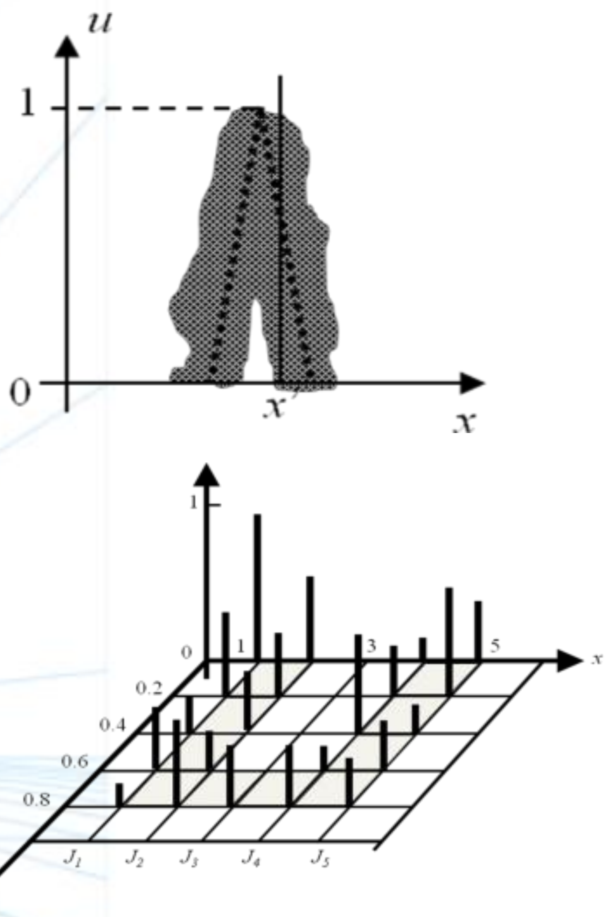
IEEE-SA Individual Members save 10% (on average)* on most standards purchased through IEEE.

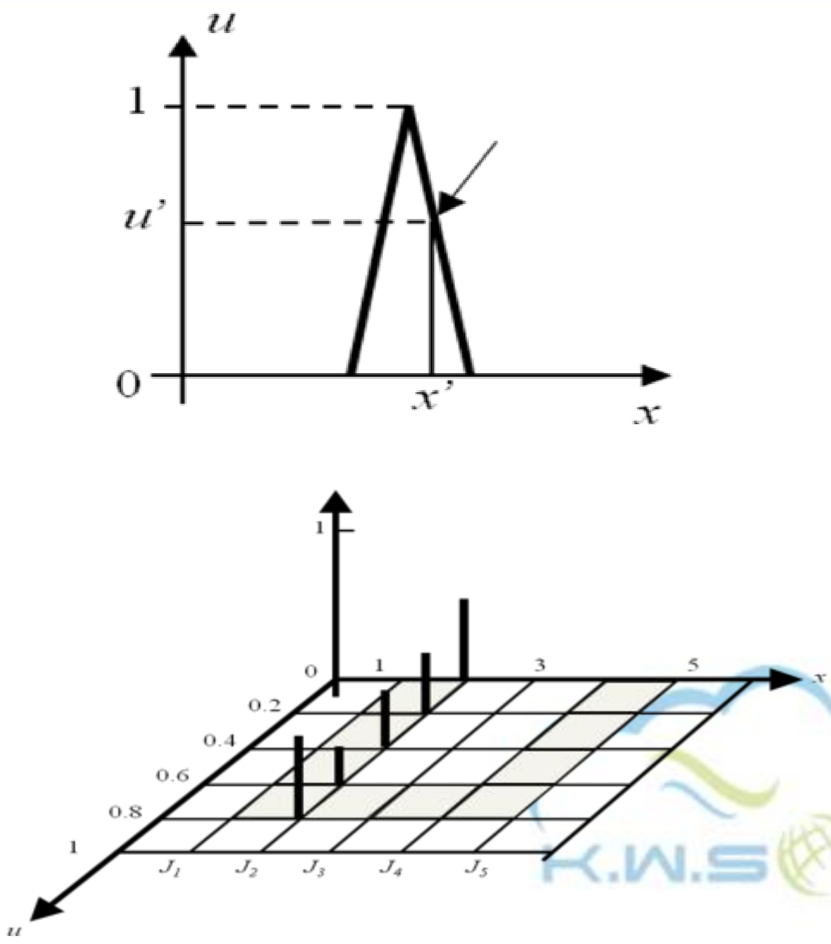
Learn More about IEEE-SA
Membership Options

** actual savings vary per standard*

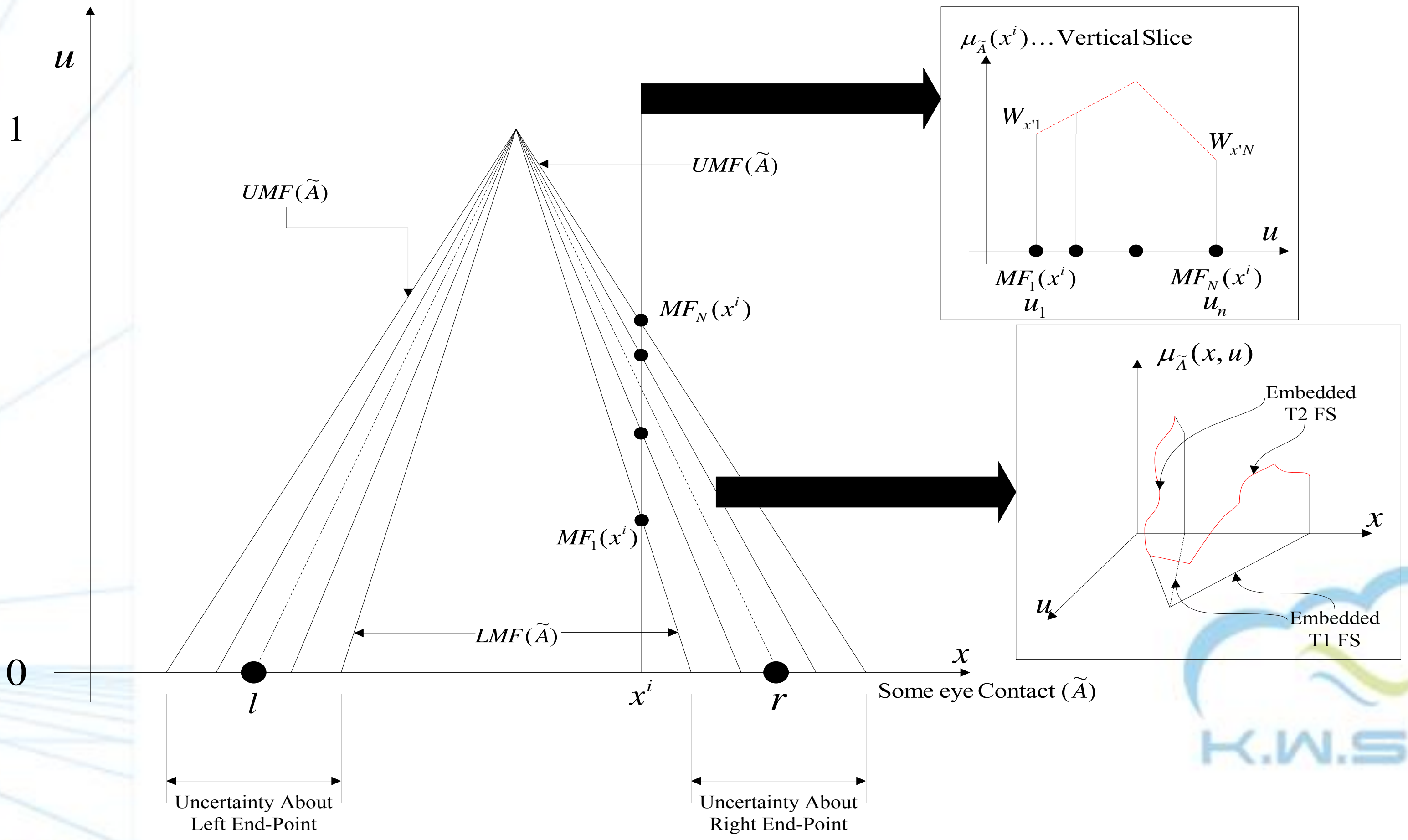
NUTN

Introduction to T2FS (1/5)

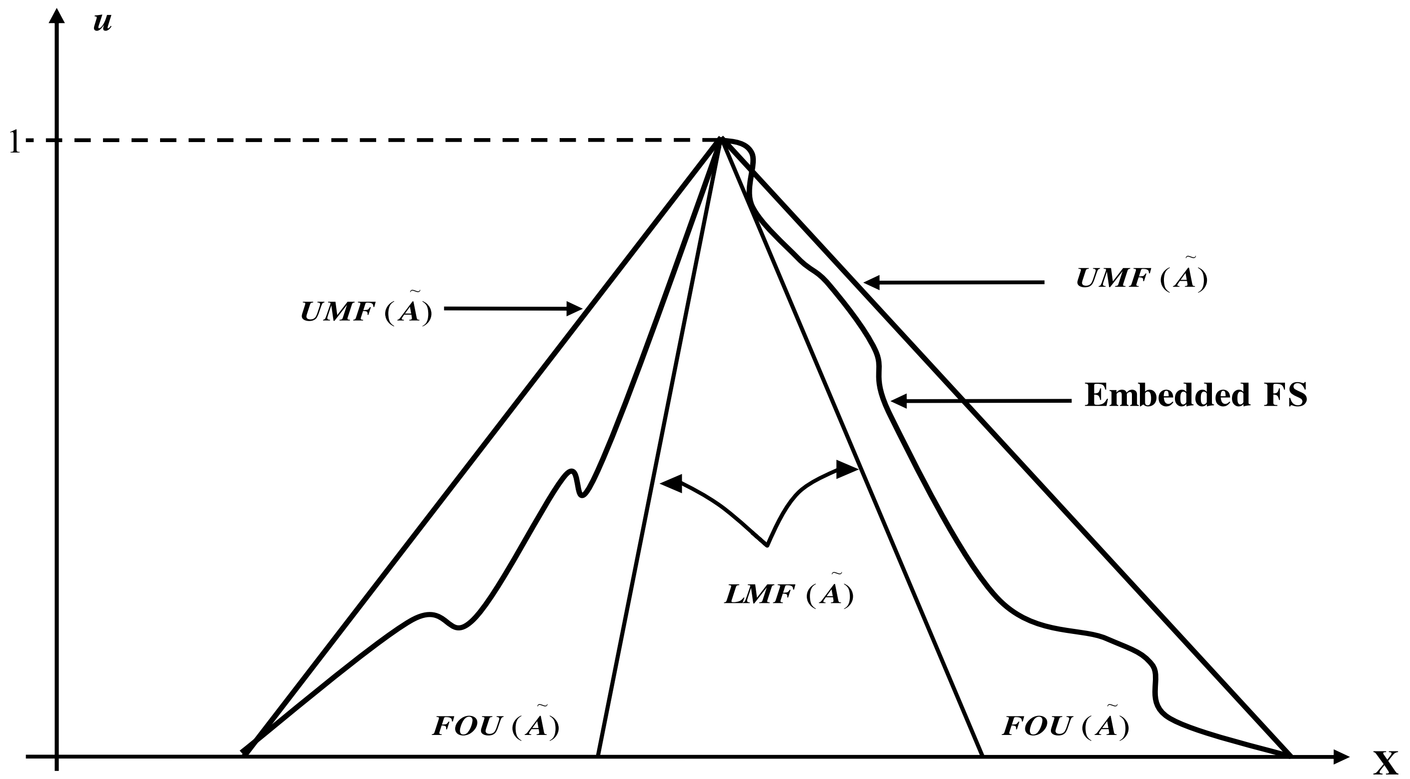




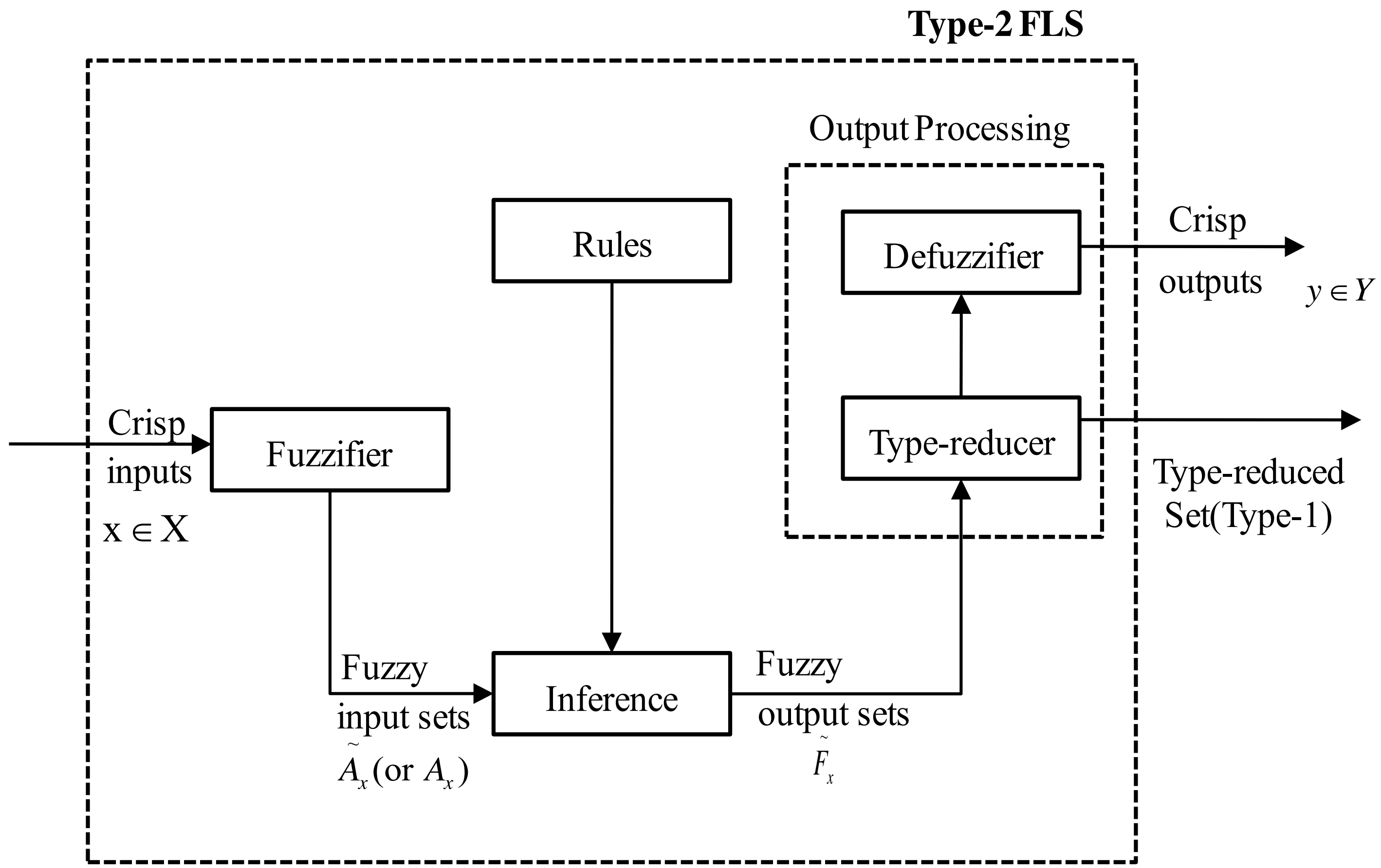
Introduction to T2FS (2/5)



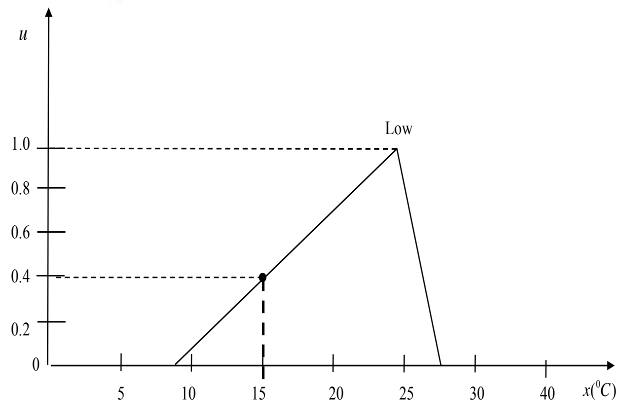
Introduction to T2FS (3/5)

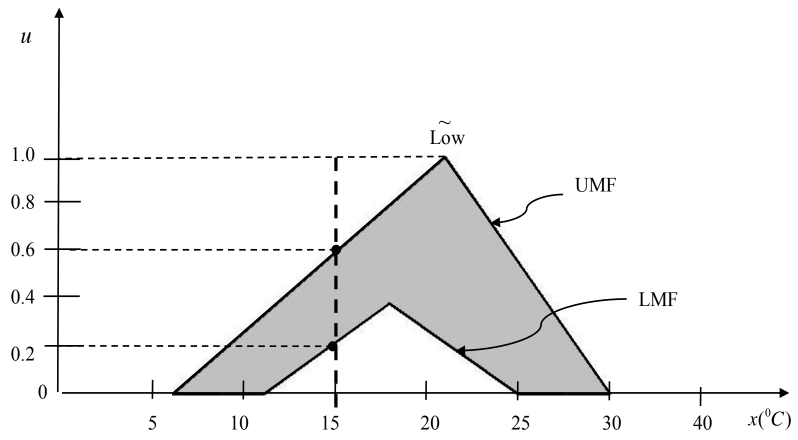


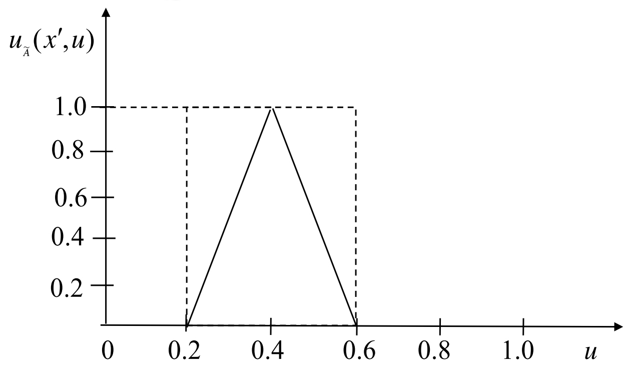
Introduction to T2FS (4/5)

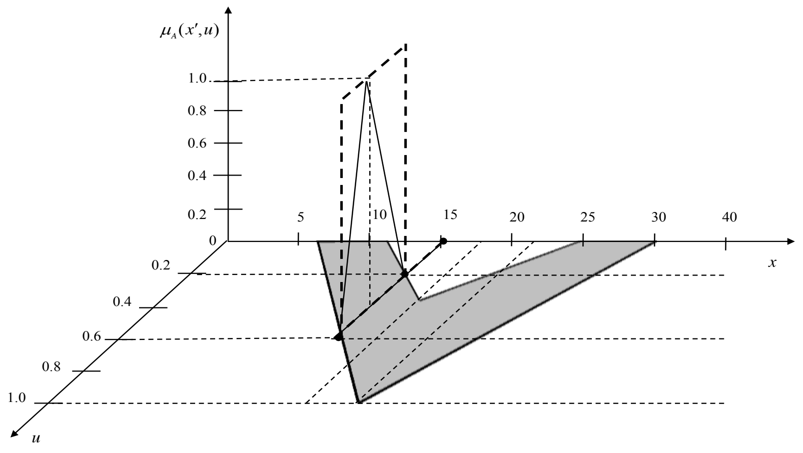


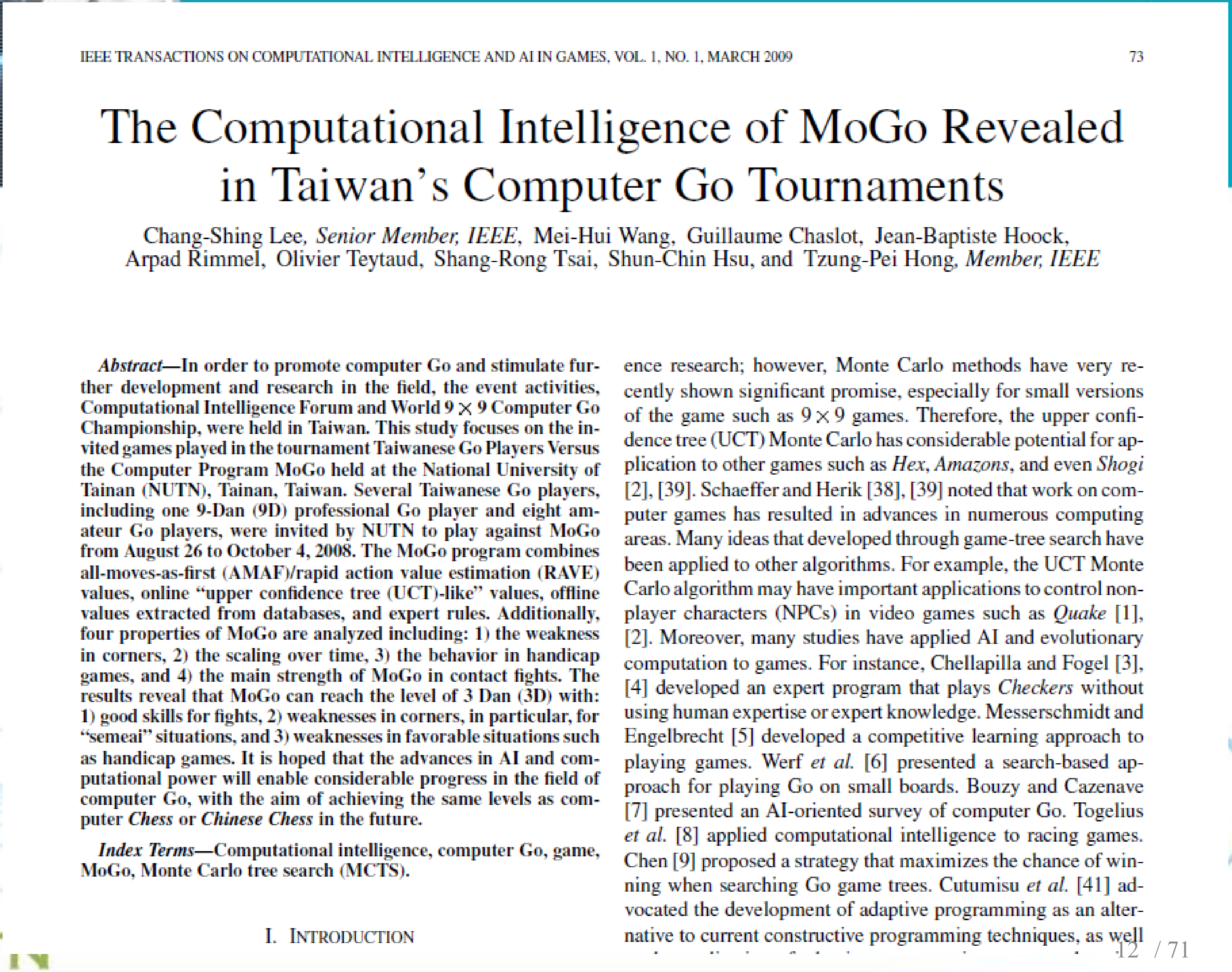
Introduction to T2FS (5/5)











IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, VOL. 1, NO. 1, MARCH 2009 73

The Computational Intelligence of MoGo Revealed in Taiwan's Computer Go Tournaments

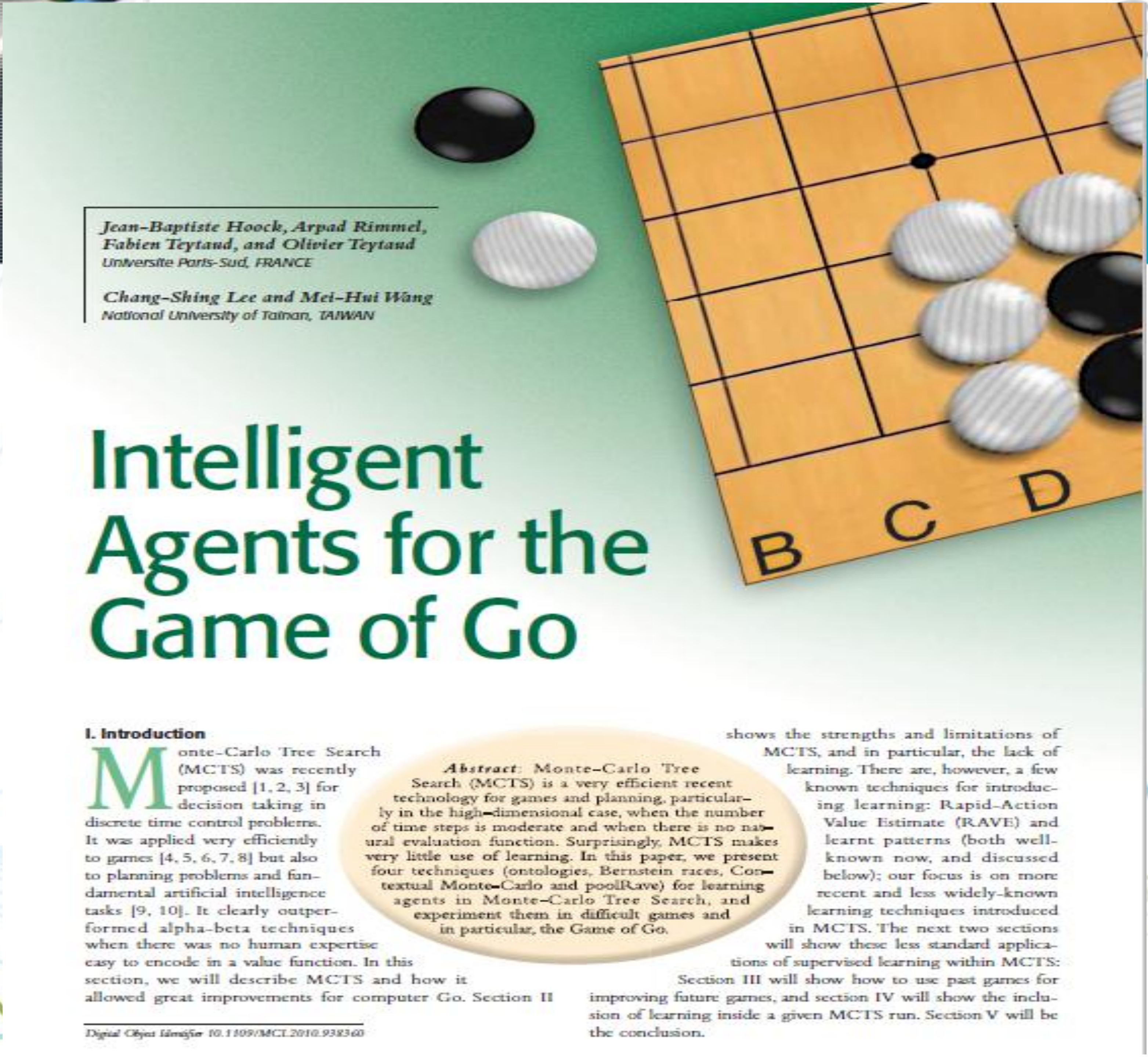
Chang-Shing Lee, *Senior Member, IEEE*, Mei-Hui Wang, Guillaume Chaslot, Jean-Baptiste Hoock, Arpad Rimmel, Olivier Teytaud, Shang-Rong Tsai, Shun-Chin Hsu, and Tzung-Pei Hong, *Member, IEEE*

***Abstract*—In order to promote computer Go and stimulate further development and research in the field, the event activities, Computational Intelligence Forum and World 9 × 9 Computer Go Championship, were held in Taiwan. This study focuses on the invited games played in the tournament Taiwanese Go Players Versus the Computer Program MoGo held at the National University of Tainan (NUTN), Tainan, Taiwan. Several Taiwanese Go players, including one 9-Dan (9D) professional Go player and eight amateur Go players, were invited by NUTN to play against MoGo from August 26 to October 4, 2008. The MoGo program combines all-moves-as-first (AMAF)/rapid action value estimation (RAVE) values, online "upper confidence tree (UCT)-like" values, offline values extracted from databases, and expert rules. Additionally, four properties of MoGo are analyzed including: 1) the weakness in corners, 2) the scaling over time, 3) the behavior in handicap games, and 4) the main strength of MoGo in contact fights. The results reveal that MoGo can reach the level of 3 Dan (3D) with: 1) good skills for fights, 2) weaknesses in corners, in particular, for "semeai" situations, and 3) weaknesses in favorable situations such as handicap games. It is hoped that the advances in AI and computational power will enable considerable progress in the field of computer Go, with the aim of achieving the same levels as computer *Chess* or *Chinese Chess* in the future.**

***Index Terms*—Computational intelligence, computer Go, game, MoGo, Monte Carlo tree search (MCTS).**

I. Introduction

ence research; however, Monte Carlo methods have very recently shown significant promise, especially for small versions of the game such as 9 × 9 games. Therefore, the upper confidence tree (UCT) Monte Carlo has considerable potential for application to other games such as *Hex*, *Amazons*, and even *Shogi* [2], [39]. Schaeffer and Herik [38], [39] noted that work on computer games has resulted in advances in numerous computing areas. Many ideas that developed through game-tree search have been applied to other algorithms. For example, the UCT Monte Carlo algorithm may have important applications to control non-player characters (NPCs) in video games such as *Quake* [1], [2]. Moreover, many studies have applied AI and evolutionary computation to games. For instance, Chellapilla and Fogel [3], [4] developed an expert program that plays *Checkers* without using human expertise or expert knowledge. Messerschmidt and Engelbrecht [5] developed a competitive learning approach to playing games. Werf *et al.* [6] presented a search-based approach for playing Go on small boards. Bouzy and Cazenave [7] presented an AI-oriented survey of computer Go. Togelius *et al.* [8] applied computational intelligence to racing games. Chen [9] proposed a strategy that maximizes the chance of winning when searching Go game trees. Cutumisu *et al.* [41] advocated the development of adaptive programming as an alternative to current constructive programming techniques, as well



Jean-Baptiste Hoock, Arpad Rimmel, Fabien Teytaud, and Olivier Teytaud
Universite Paris-Sud, FRANCE

Chang-Shing Lee and Mei-Hui Wang
National University of Tainan, TAIWAN

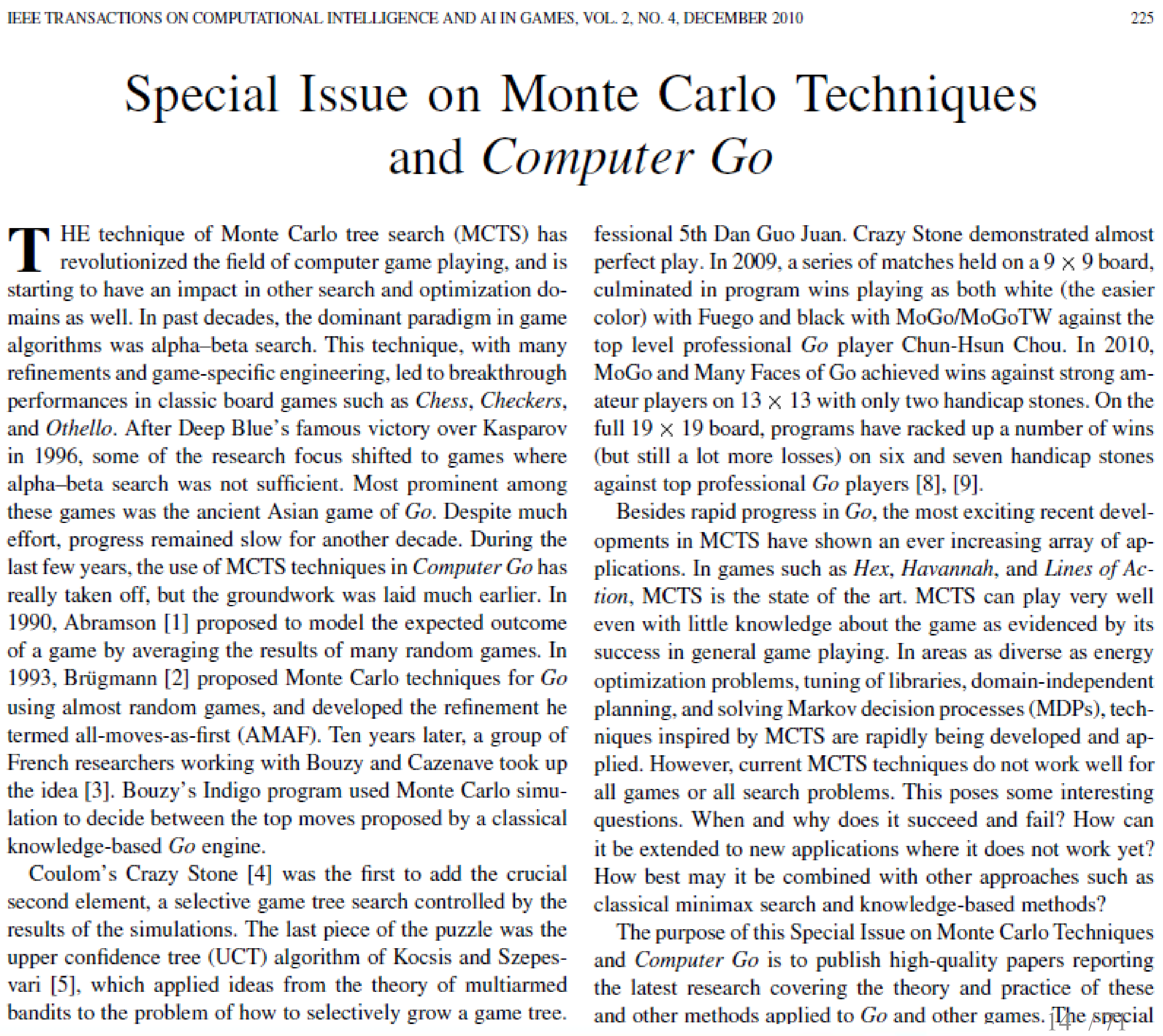
Intelligent Agents for the Game of Go

Abstract: Monte-Carlo Tree Search (MCTS) is a very efficient recent technology for games and planning, particularly in the high-dimensional case, when the number of time steps is moderate and when there is no natural evaluation function. Surprisingly, MCTS makes very little use of learning. In this paper, we present four techniques (ontologies, Bernstein races, Contextual Monte-Carlo and poolRave) for learning agents in Monte-Carlo Tree Search, and experiment them in difficult games and in particular, the Game of Go.

I. Introduction

Monte-Carlo Tree Search (MCTS) was recently proposed [1, 2, 3] for decision taking in discrete time control problems. It was applied very efficiently to games [4, 5, 6, 7, 8] but also to planning problems and fundamental artificial intelligence tasks [9, 10]. It clearly outperformed alpha-beta techniques when there was no human expertise easy to encode in a value function. In this section, we will describe MCTS and how it allowed great improvements for computer Go. Section II shows the strengths and limitations of MCTS, and in particular, the lack of learning. There are, however, a few known techniques for introducing learning: Rapid-Action Value Estimate (RAVE) and learnt patterns (both well-known now, and discussed below); our focus is on more recent and less widely-known learning techniques introduced in MCTS. The next two sections will show these less standard applications of supervised learning within MCTS: Section III will show how to use past games for improving future games, and section IV will show the inclusion of learning inside a given MCTS run. Section V will be the conclusion.

Digital Object Identifier 10.1109/MCI.2010.938360



IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES, VOL. 2, NO. 4, DECEMBER 2010

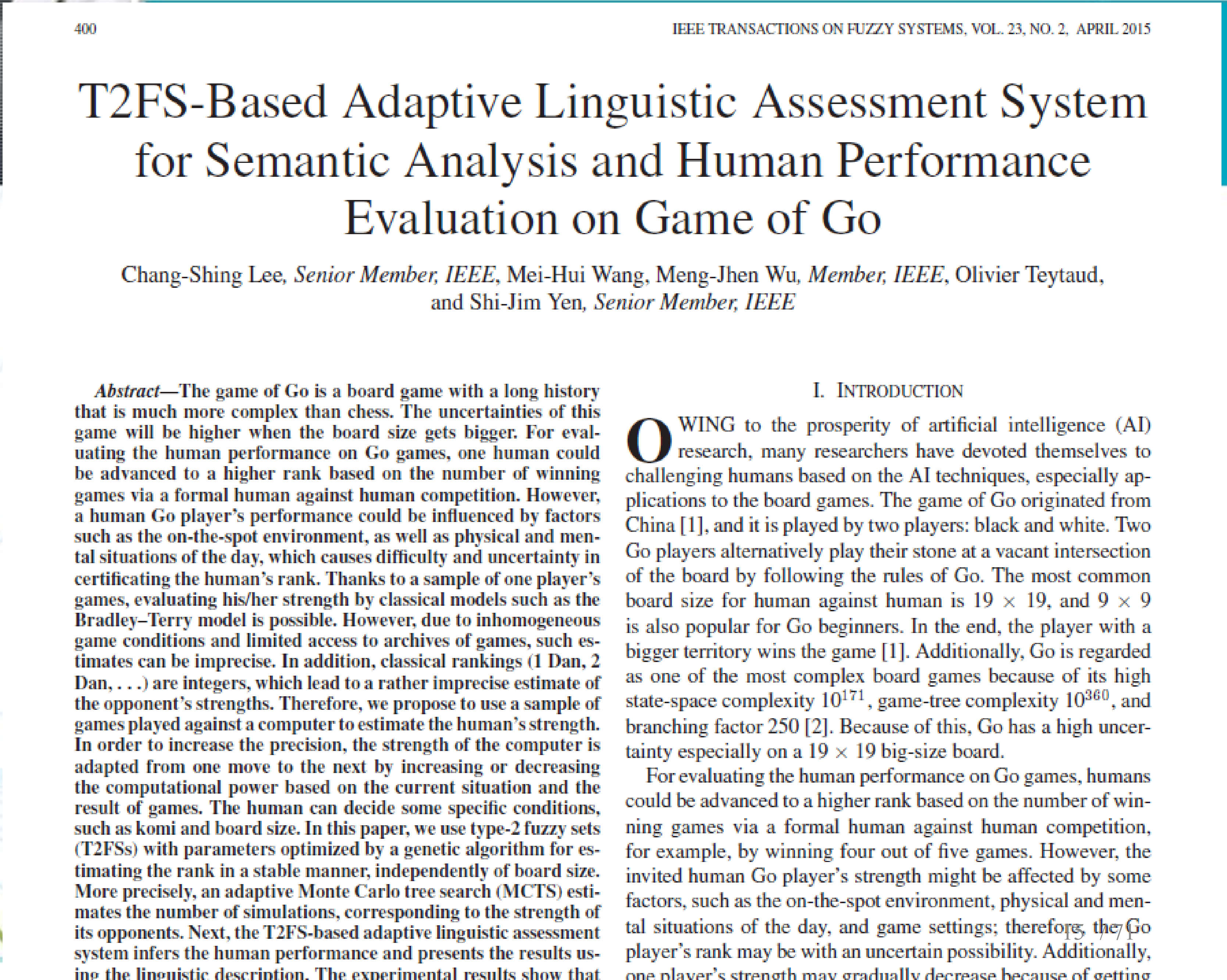
Special Issue on Monte Carlo Techniques and *Computer Go*

THE technique of Monte Carlo tree search (MCTS) has revolutionized the field of computer game playing, and is starting to have an impact in other search and optimization domains as well. In past decades, the dominant paradigm in game algorithms was alpha–beta search. This technique, with many refinements and game-specific engineering, led to breakthrough performances in classic board games such as *Chess*, *Checkers*, and *Othello*. After Deep Blue's famous victory over Kasparov in 1996, some of the research focus shifted to games where alpha–beta search was not sufficient. Most prominent among these games was the ancient Asian game of *Go*. Despite much effort, progress remained slow for another decade. During the last few years, the use of MCTS techniques in *Computer Go* has really taken off, but the groundwork was laid much earlier. In 1990, Abramson [1] proposed to model the expected outcome of a game by averaging the results of many random games. In 1993, Brügmann [2] proposed Monte Carlo techniques for *Go* using almost random games, and developed the refinement he termed all-moves-as-first (AMAF). Ten years later, a group of French researchers working with Bouzy and Cazenave took up the idea [3]. Bouzy's Indigo program used Monte Carlo simulation to decide between the top moves proposed by a classical knowledge-based *Go* engine.

Coulom's Crazy Stone [4] was the first to add the crucial second element, a selective game tree search controlled by the results of the simulations. The last piece of the puzzle was the upper confidence tree (UCT) algorithm of Kocsis and Szepesvari [5], which applied ideas from the theory of multiarmed bandits to the problem of how to selectively grow a game tree. fessional 5th Dan Guo Juan. Crazy Stone demonstrated almost perfect play. In 2009, a series of matches held on a 9×9 board, culminated in program wins playing as both white (the easier color) with Fuego and black with MoGo/MoGoTW against the top level professional *Go* player Chun-Hsun Chou. In 2010, MoGo and Many Faces of Go achieved wins against strong amateur players on 13×13 with only two handicap stones. On the full 19×19 board, programs have racked up a number of wins (but still a lot more losses) on six and seven handicap stones against top professional *Go* players [8], [9].

Besides rapid progress in *Go*, the most exciting recent developments in MCTS have shown an ever increasing array of applications. In games such as *Hex*, *Havannah*, and *Lines of Action*, MCTS is the state of the art. MCTS can play very well even with little knowledge about the game as evidenced by its success in general game playing. In areas as diverse as energy optimization problems, tuning of libraries, domain-independent planning, and solving Markov decision processes (MDPs), techniques inspired by MCTS are rapidly being developed and applied. However, current MCTS techniques do not work well for all games or all search problems. This poses some interesting questions. When and why does it succeed and fail? How can it be extended to new applications where it does not work yet? How best may it be combined with other approaches such as classical minimax search and knowledge-based methods?

The purpose of this Special Issue on Monte Carlo Techniques and *Computer Go* is to publish high-quality papers reporting the latest research covering the theory and practice of these and other methods applied to *Go* and other games. The special



T2FS-Based Adaptive Linguistic Assessment System for Semantic Analysis and Human Performance Evaluation on Game of Go

Chang-Shing Lee, *Senior Member, IEEE*, Mei-Hui Wang, Meng-Jhen Wu, *Member, IEEE*, Olivier Teytaud, and Shi-Jim Yen, *Senior Member, IEEE*

***Abstract*—The game of Go is a board game with a long history that is much more complex than chess. The uncertainties of this game will be higher when the board size gets bigger. For evaluating the human performance on Go games, one human could be advanced to a higher rank based on the number of winning games via a formal human against human competition. However, a human Go player's performance could be influenced by factors such as the on-the-spot environment, as well as physical and mental situations of the day, which causes difficulty and uncertainty in certificating the human's rank. Thanks to a sample of one player's games, evaluating his/her strength by classical models such as the Bradley–Terry model is possible. However, due to inhomogeneous game conditions and limited access to archives of games, such estimates can be imprecise. In addition, classical rankings (1 Dan, 2 Dan, . . .) are integers, which lead to a rather imprecise estimate of the opponent's strengths. Therefore, we propose to use a sample of games played against a computer to estimate the human's strength. In order to increase the precision, the strength of the computer is adapted from one move to the next by increasing or decreasing the computational power based on the current situation and the result of games. The human can decide some specific conditions, such as komi and board size. In this paper, we use type-2 fuzzy sets (T2FSs) with parameters optimized by a genetic algorithm for estimating the rank in a stable manner, independently of board size. More precisely, an adaptive Monte Carlo tree search (MCTS) estimates the number of simulations, corresponding to the strength of its opponents. Next, the T2FS-based adaptive linguistic assessment system infers the human performance and presents the results using the linguistic description. The experimental results show that**

I. Introduction

OWING to the prosperity of artificial intelligence (AI) research, many researchers have devoted themselves to challenging humans based on the AI techniques, especially applications to the board games. The game of Go originated from China [1], and it is played by two players: black and white. Two Go players alternatively play their stone at a vacant intersection of the board by following the rules of Go. The most common board size for human against human is 19×19, and 9×9 is also popular for Go beginners. In the end, the player with a bigger territory wins the game [1]. Additionally, Go is regarded as one of the most complex board games because of its high state-space complexity 10^{171}, game-tree complexity 10^{360}, and branching factor 250 [2]. Because of this, Go has a high uncertainty especially on a 19×19 big-size board.

For evaluating the human performance on Go games, humans could be advanced to a higher rank based on the number of winning games via a formal human against human competition, for example, by winning four out of five games. However, the invited human Go player's strength might be affected by some factors, such as the on-the-spot environment, physical and mental situations of the day, and game settings; therefore, the Go player's rank may be with an uncertain possibility. Additionally, one player's strength may gradually decrease because of getting

Discussion Forum

Chang-Shing Lee and Mei-Hui Wang
Department of Computer Science and Information Engineering, National University of Tainan, *TAIWAN*

Shi-Jim Yen
Department of Computer Science and Information Engineering, National Dong Hwa University, *TAIWAN*

Ting-Han Wei and I-Chen Wu
Department of Computer Science, National Chiao Tung University, *TAIWAN*

Ping-Chiang Chou and Chun-Hsun Chou
Taiwan Go Association, *TAIWAN*

Ming-Wan Wang
Nihon Ki-in Go Institute, *JAPAN*

Tai-Hsiung Yang
Haifong Weiqi Academy, *TAIWAN*

Human vs. Computer Go: Review and Prospect

Abstract
The Google DeepMind challenge match in March 2016 was a historic achievement for computer Go development. This article discusses the development of computational intelligence (CI) and its relative strength in comparison with human intelligence for the game of Go. We first summarize the milestones achieved for computer Go from 1998 to 2016. Then, the computer Go programs that have participated in previous IEEE CIS competitions as well as methods and techniques used in AlphaGo are briefly introduced. Commentaries from three high-level professional Go players on the five AlphaGo versus Lee

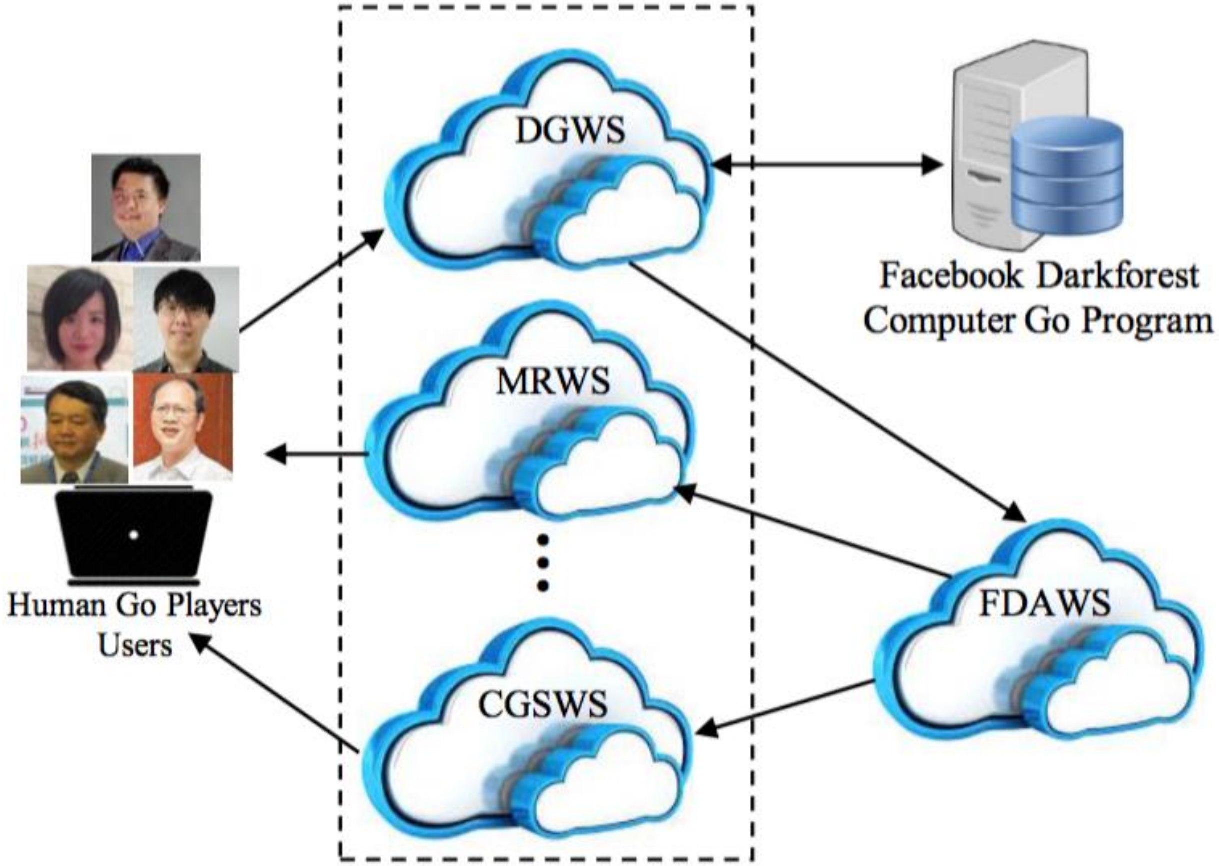
IMAGE LICENSED BY GRAPHIC STOCK

puter Go competitions in IEEE CIS flagship conferences since 2009. Fig. 1 shows the year and the location of the confer-

Professional Go players are ranked entirely in dan, abbreviated with the letter *P* (e.g. Lee Sedol is ranked at 9P). In the amateur level, each difference in rank roughly translates to a single stone of *handicap* (H), where the weaker player is allowed to place an additional stone on the board prior to play to even out the game. The skill difference between professional ranks is much less than one stone for every rank difference. Go is typically played on 19×19 size boards, but 9×9 size boards are also common for beginners. The complexity of the 9×9 game is far less than the standard game, and the 9×9 game had been one of the interim goals for computer Go programs. Go is a game that

NUTN

Dynamic Assessment and IRT-based Learning Application



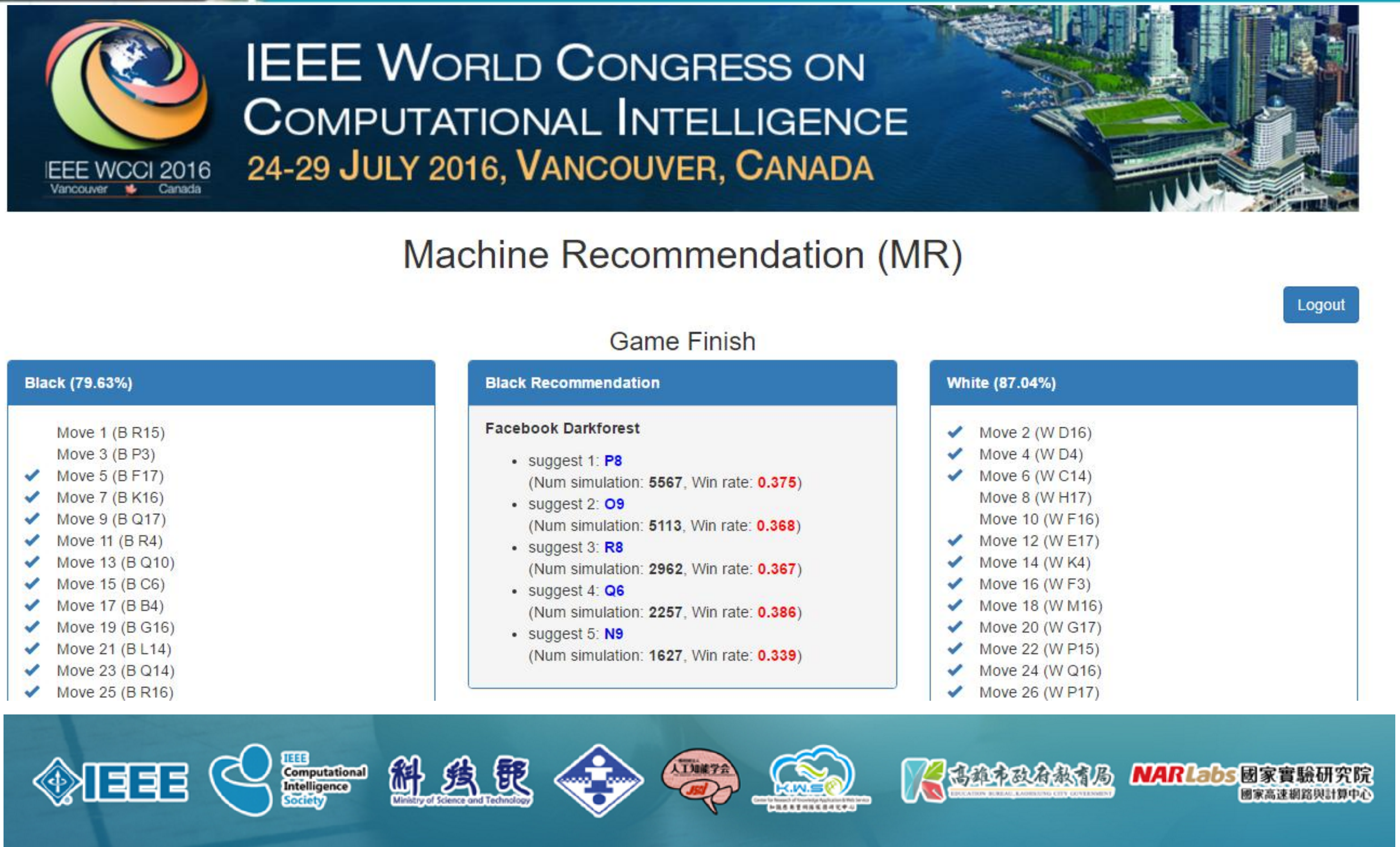
Dynamic Assessment and IRT-based Learning Application

IEEE World Congress on Computational Intelligence

24-29 July 2016, Vancouver, Canada

Machine Recommendation (MR)

Please Sign In



IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE
24-29 JULY 2016, VANCOUVER, CANADA

IEEE WCCI 2016
Vancouver Canada

Machine Recommendation (MR)

Logout

Game Finish

Black (79.63%)

- Move 1 (B R15)
- Move 3 (B P3)
- ✓ Move 5 (B F17)
- ✓ Move 7 (B K16)
- ✓ Move 9 (B Q17)
- ✓ Move 11 (B R4)
- ✓ Move 13 (B Q10)
- ✓ Move 15 (B C6)
- ✓ Move 17 (B B4)
- ✓ Move 19 (B G16)
- ✓ Move 21 (B L14)
- ✓ Move 23 (B Q14)
- ✓ Move 25 (B R16)

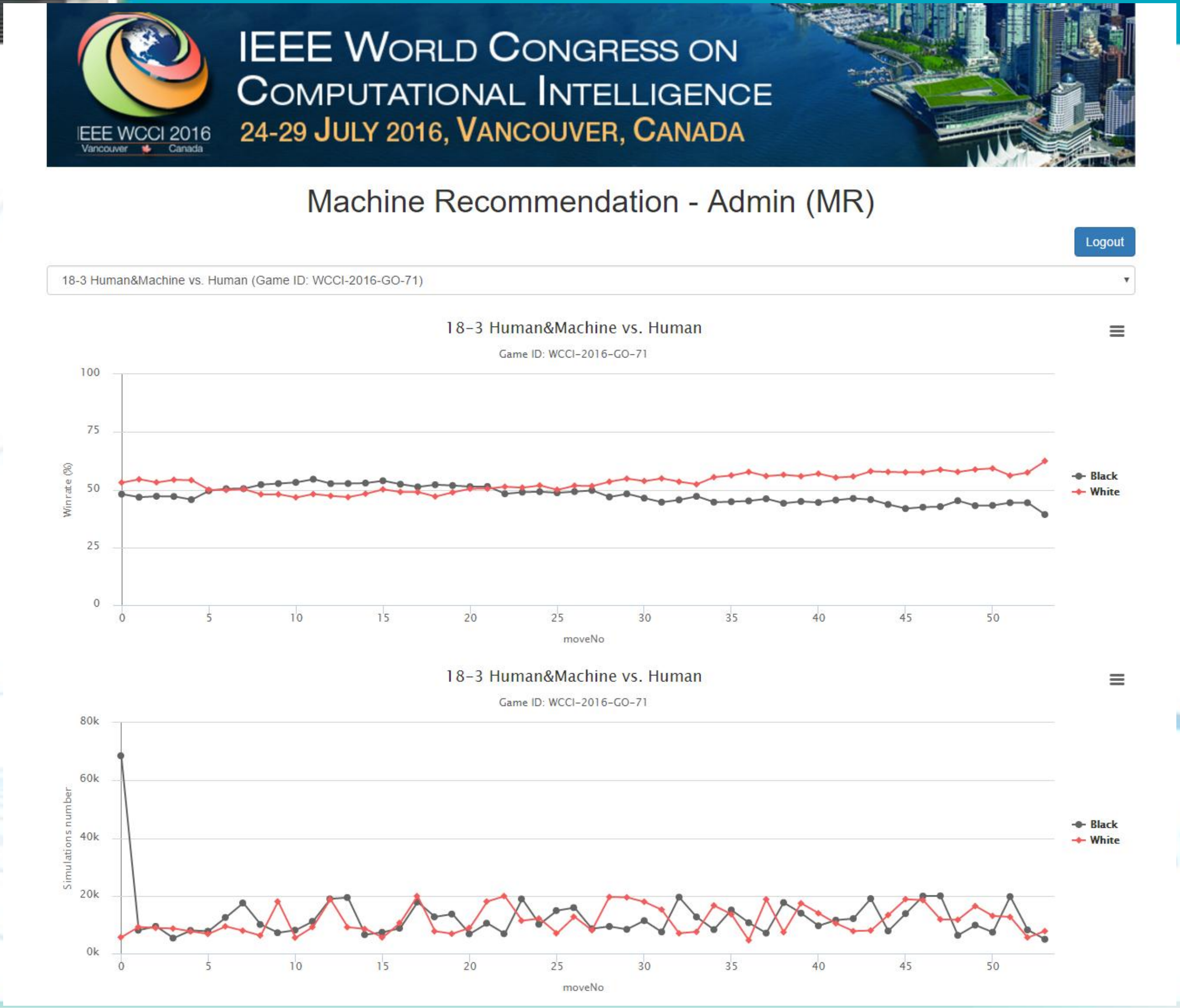
Black Recommendation

Facebook Darkforest

- suggest 1: **P8**
  (Num simulation: **5567**, Win rate: **0.375**)
- suggest 2: **O9**
  (Num simulation: **5113**, Win rate: **0.368**)
- suggest 3: **R8**
  (Num simulation: **2962**, Win rate: **0.367**)
- suggest 4: **Q6**
  (Num simulation: **2257**, Win rate: **0.386**)
- suggest 5: **N9**
  (Num simulation: **1627**, Win rate: **0.339**)

White (87.04%)

- ✓ Move 2 (W D16)
- ✓ Move 4 (W D4)
- ✓ Move 6 (W C14)
- Move 8 (W H17)
- Move 10 (W F16)
- ✓ Move 12 (W E17)
- ✓ Move 14 (W K4)
- ✓ Move 16 (W F3)
- ✓ Move 18 (W M16)
- ✓ Move 20 (W G17)
- ✓ Move 22 (W P15)
- ✓ Move 24 (W Q16)
- ✓ Move 26 (W P17)



IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE

24-29 JULY 2016, VANCOUVER, CANADA

Machine Recommendation - Admin (MR)

Logout

18-3 Human&Machine vs. Human (Game ID: WCCI-2016-GO-71)

Winrate Difference

Move 1 (局勢尚難確定)
Move 2 (局勢尚難確定)
Move 3 (局勢尚難確定)
Move 4 (局勢尚難確定)
Move 5 (局勢尚難確定)
Move 6 (局勢尚難確定)
Move 7 (局勢尚難確定)
Move 8 (局勢尚難確定)
Move 9 (局勢尚難確定)
Move 10 (局勢尚難確定)
Move 11 (局勢尚難確定)
Move 12 (局勢尚難確定)
Move 13 (局勢尚難確定)
Move 14 (局勢尚難確定)
Move 15 (局勢尚難確定)
Move 16 (局勢尚難確定)
Move 17 (局勢尚難確定)
Move 18 (局勢尚難確定)
Move 19 (局勢尚難確定)
Move 20 (局勢尚難確定)
Move 21 (局勢尚難確定)
Move 22 (局勢尚難確定)
Move 23 (局勢尚難確定)
Move 24 (局勢尚難確定)
Move 25 (局勢尚難確定)
Move 26 (局勢尚難確定)
Move 27 (局勢尚難確定)
Move 28 (局勢尚難確定)
Move 29

Simulation Difference

Move 1 (局勢尚難確定)
Move 2 (局勢尚難確定)
Move 3 (局勢尚難確定)
Move 4 (局勢尚難確定)
Move 5 (局勢尚難確定)
Move 6 (局勢尚難確定)
Move 7 (局勢尚難確定)
Move 8 (局勢尚難確定)
Move 9 (局勢尚難確定)
Move 10 (局勢尚難確定)
Move 11 (局勢尚難確定)
Move 12 (局勢尚難確定)
Move 13 (白棋明顯優勢)
Move 14 (黑棋可能優勢)
Move 15 (白棋明顯優勢)
Move 16 (黑棋明顯優勢)
Move 17 (白棋可能優勢)
Move 18 (黑棋可能優勢)
Move 19 (局勢尚難確定)
Move 20 (白棋明顯優勢)
Move 21 (黑棋明顯優勢)
Move 22 (黑棋可能優勢)
Move 23 (白棋明顯優勢)
Move 24 (局勢尚難確定)
Move 25 (白棋明顯優勢)
Move 26 (局勢尚難確定)
Move 27 (局勢尚難確定)
Move 28 (黑棋明顯優勢)
Move 29

FML Assessment-1

Move 1 (局勢尚難確定)
Move 2 (局勢尚難確定)
Move 3 (局勢尚難確定)
Move 4 (局勢尚難確定)
Move 5 (局勢尚難確定)
Move 6 (局勢尚難確定)
Move 7 (局勢尚難確定)
Move 8 (局勢尚難確定)
Move 9 (局勢尚難確定)
Move 10 (局勢尚難確定)
Move 11 (白棋可能優勢)
Move 12 (白棋可能優勢)
Move 13 (白棋明顯優勢)
Move 14 (白棋明顯優勢)
Move 15 (白棋明顯優勢)
Move 16 (白棋明顯優勢)
Move 17 (白棋可能優勢)
Move 18 (局勢尚難確定)
Move 19 (局勢尚難確定)
Move 20 (白棋明顯優勢)
Move 21 (白棋明顯優勢)
Move 22 (局勢尚難確定)
Move 23 (局勢尚難確定)
Move 24 (白棋可能優勢)
Move 25 (白棋明顯優勢)
Move 26 (白棋明顯優勢)
Move 27 (白棋明顯優勢)
Move 28 (白棋明顯優勢)
Move 29

FML Assessment-2

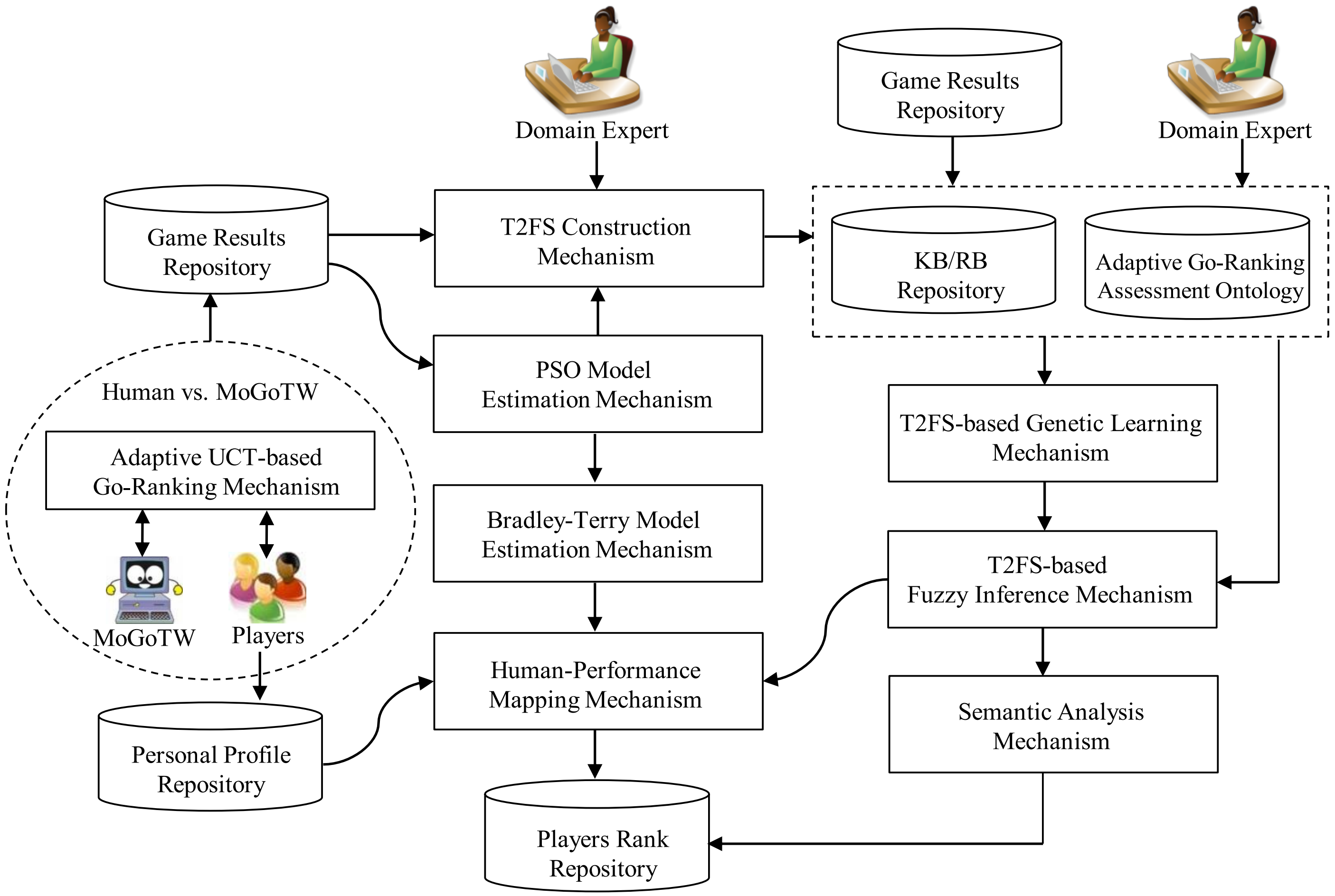
Move 1 (局勢尚難確定)
Move 2 (局勢尚難確定)
Move 3 (局勢尚難確定)
Move 4 (局勢尚難確定)
Move 5 (局勢尚難確定)
Move 6 (局勢尚難確定)
Move 7 (局勢尚難確定)
Move 8 (局勢尚難確定)
Move 9 (局勢尚難確定)
Move 10 (局勢尚難確定)
Move 11 (局勢尚難確定)
Move 12 (局勢尚難確定)
Move 13 (局勢尚難確定)
Move 14 (局勢尚難確定)
Move 15 (黑棋可能優勢)
Move 16 (黑棋可能優勢)
Move 17 (局勢尚難確定)
Move 18 (局勢尚難確定)
Move 19 (局勢尚難確定)
Move 20 (白棋可能優勢)
Move 21 (白棋可能優勢)
Move 22 (局勢尚難確定)
Move 23 (局勢尚難確定)
Move 24 (局勢尚難確定)
Move 25 (黑棋可能優勢)
Move 26 (局勢尚難確定)
Move 27 (局勢尚難確定)
Move 28 (黑棋可能優勢)
Move 29

Video Demonstration

- Taiwan Open 2009
- Human vs. Computer Go @ IEEE WCCI 2012
- Human vs. Computer Go @ FUZZ-IEEE 2011
- Human vs. Computer Go in Taiwan in 2011

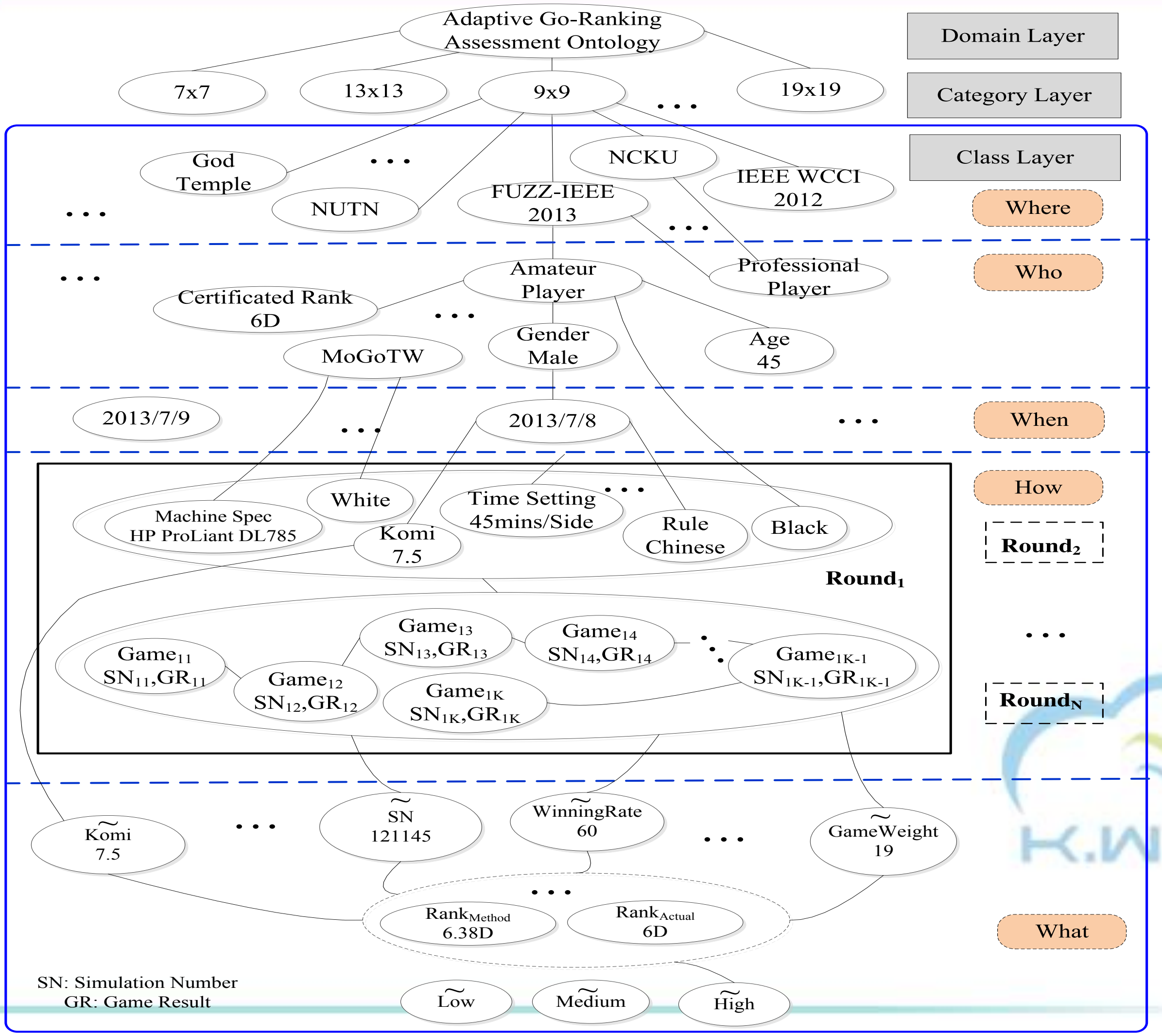
NUTN

Adaptive linguistic assessment


Domain Expert
Game Results Repository
Domain Expert
Game Results Repository
T2FS Construction Mechanism
KB/RB Repository
Adaptive Go-Ranking Assessment Ontology
PSO Model Estimation Mechanism
Human vs. MoGoTW
Adaptive UCT-based Go-Ranking Mechanism
MoGoTW
Players
T2FS-based Genetic Learning Mechanism
Bradley-Terry Model Estimation Mechanism
T2FS-based Fuzzy Inference Mechanism
Human-Performance Mapping Mechanism
Semantic Analysis Mechanism
Personal Profile Repository
Players Rank Repository

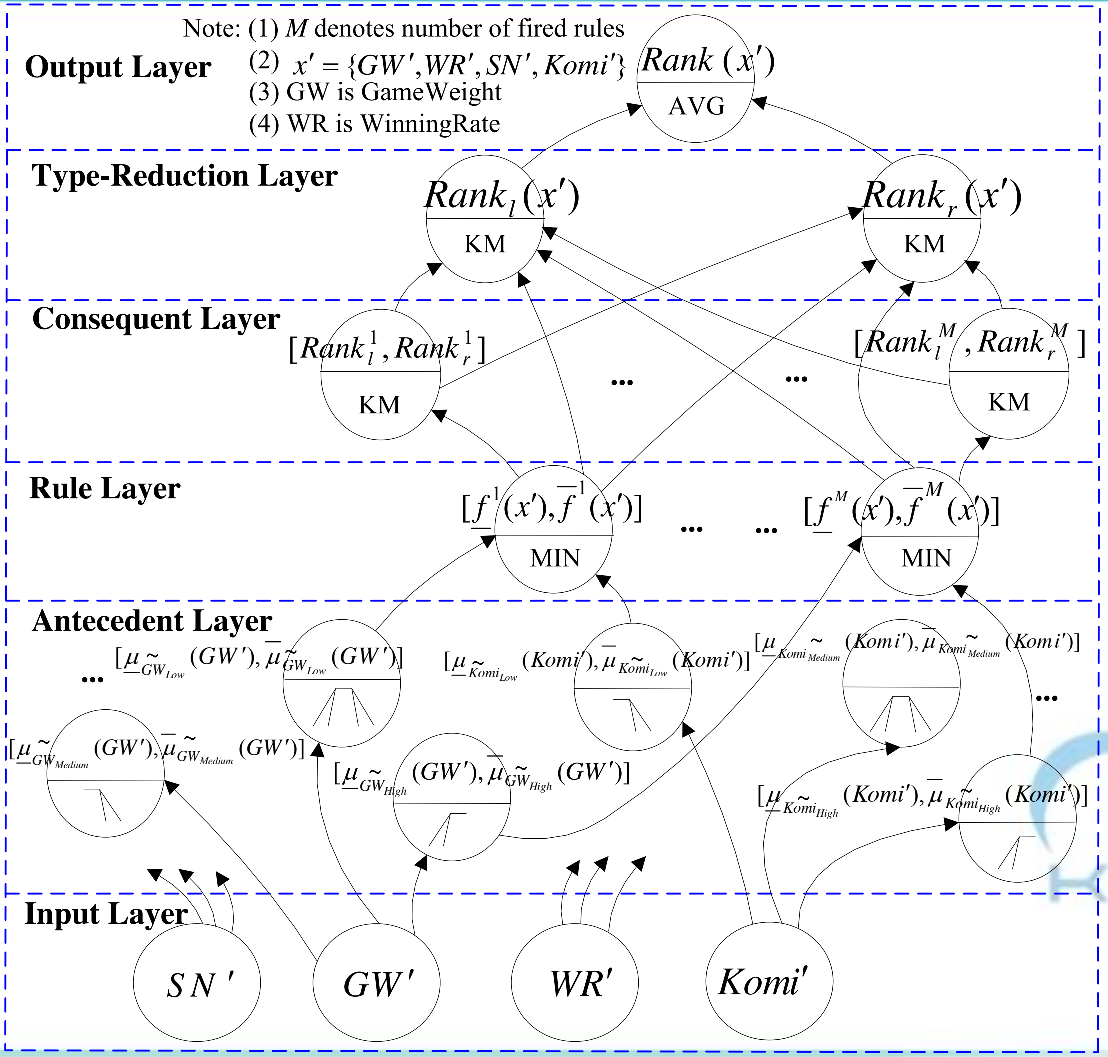
NUTN

Go-ranking assessment ontology



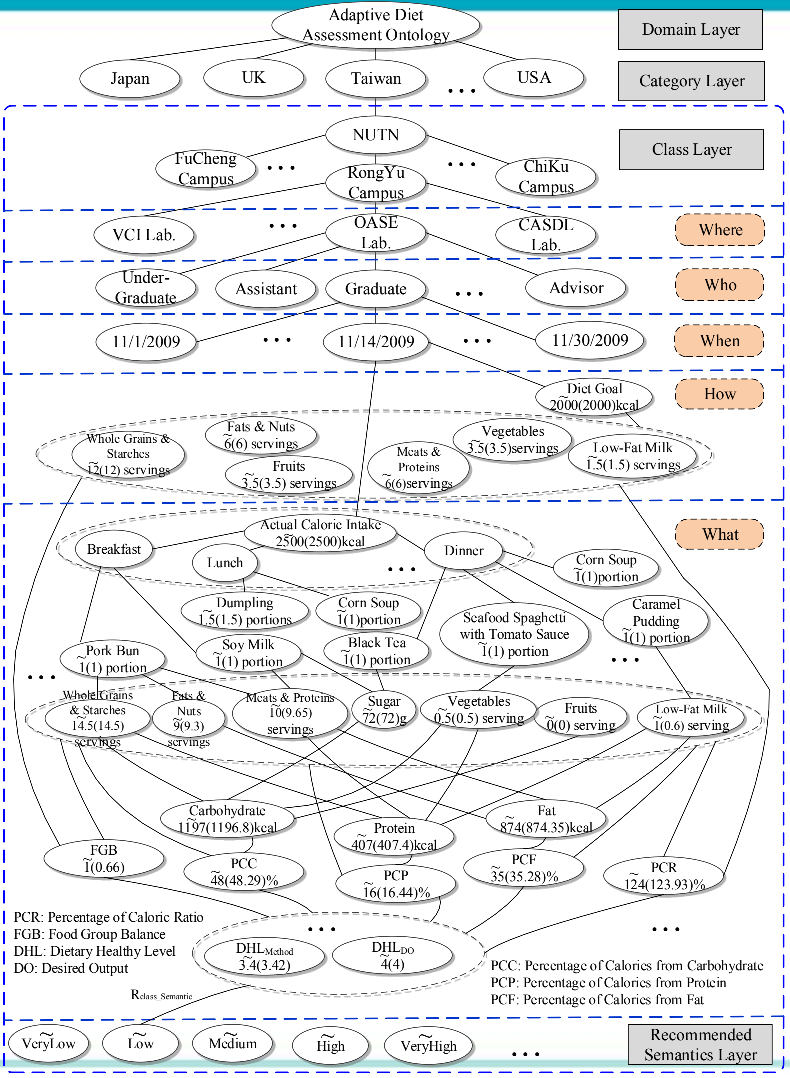
NUTN

Fuzzy inference structure

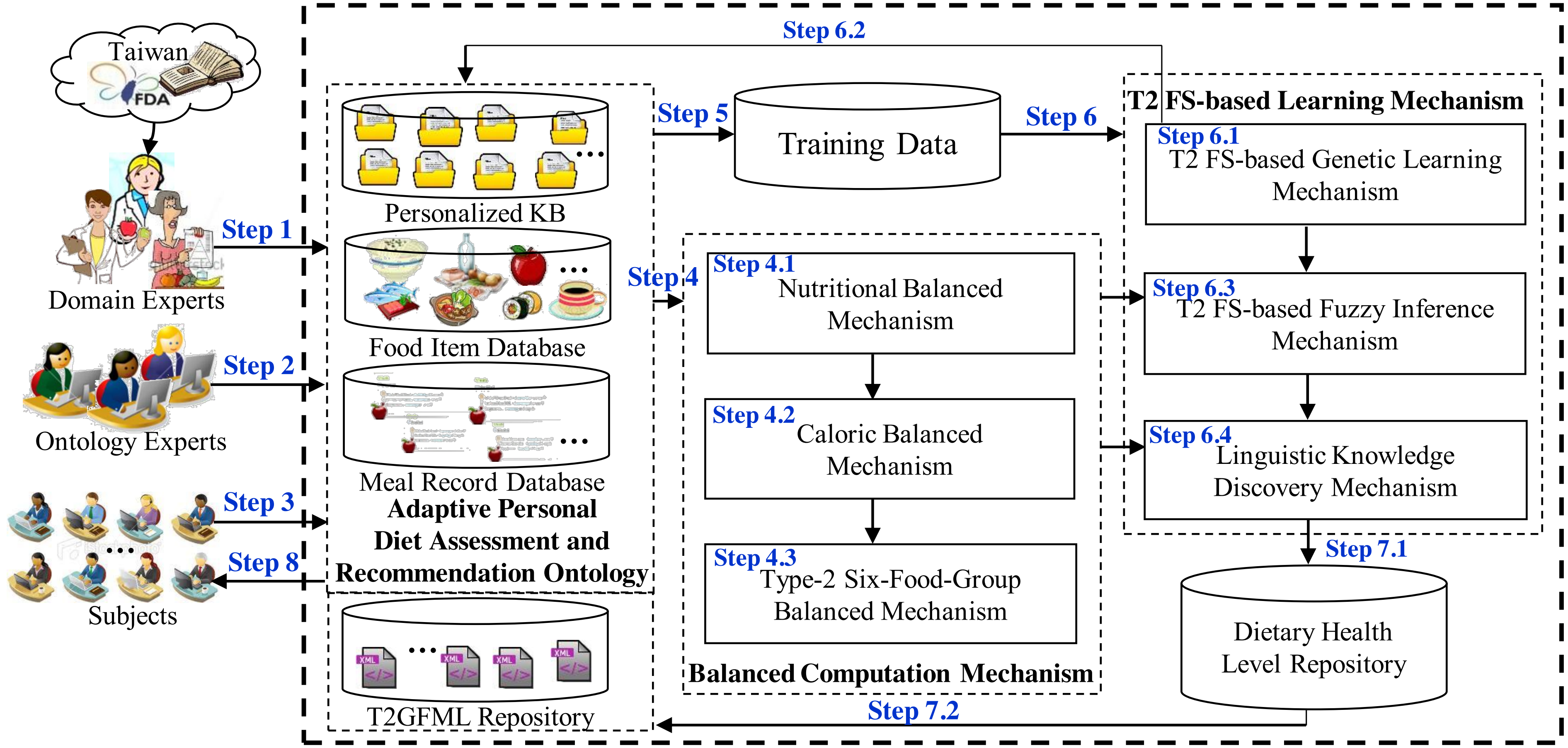

Output Layer
Note: (1) M denotes number of fired rules
(2) $x' = \{GW', WR', SN', Komi'\}$
(3) GW is GameWeight
(4) WR is WinningRate
$Rank(x')$
AVG
Type-Reduction Layer
$Rank_l(x')$
KM
$Rank_r(x')$
KM
Consequent Layer
$[Rank_l^1, Rank_r^1]$
KM
...
...
$[Rank_l^M, Rank_r^M]$
KM
Rule Layer
$[\underline{f}^1(x'), \overline{f}^1(x')]$
MIN
... ...
$[\underline{f}^M(x'), \overline{f}^M(x')]$
MIN
Antecedent Layer
...
$[\underline{\mu}_{\tilde{GW}_{Low}}(GW'), \overline{\mu}_{\tilde{GW}_{Low}}(GW')]$
$[\underline{\mu}_{\tilde{Komi}_{Low}}(Komi'), \overline{\mu}_{\tilde{Komi}_{Low}}(Komi')]$
$[\underline{\mu}_{\tilde{Komi}_{Medium}}(Komi'), \overline{\mu}_{\tilde{Komi}_{Medium}}(Komi')]$
...
$[\underline{\mu}_{\tilde{GW}_{Medium}}(GW'), \overline{\mu}_{\tilde{GW}_{Medium}}(GW')]$
$[\underline{\mu}_{\tilde{GW}_{High}}(GW'), \overline{\mu}_{\tilde{GW}_{High}}(GW')]$
$[\underline{\mu}_{\tilde{Komi}_{High}}(Komi'), \overline{\mu}_{\tilde{Komi}_{High}}(Komi')]$
Input Layer
SN'
GW'
WR'
$Komi'$

Personalized Diet Recommendation

Diet assessment / recommendation ontology

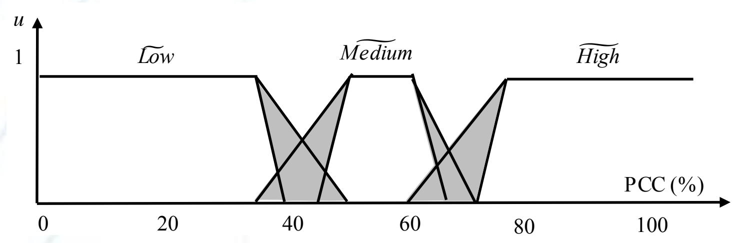

Adaptive Diet Assessment Ontology
Domain Layer
Japan
UK
Taiwan
USA
Category Layer
NUTN
FuCheng Campus
RongYu Campus
ChiKu Campus
Class Layer
VCI Lab.
OASE Lab.
CASDL Lab.
Where
Under-Graduate
Assistant
Graduate
Advisor
Who
11/1/2009
11/14/2009
11/30/2009
When
Diet Goal 2000(2000)kcal
How
Whole Grains & Starches 12(12) servings
Fats & Nuts 6(6) servings
Fruits 3.5(3.5) servings
Meats & Proteins 6(6)servings
Vegetables 3.5(3.5)servings
Low-Fat Milk 1.5(1.5) servings
Actual Caloric Intake 2500(2500)kcal
What
Breakfast
Lunch
Dinner
Corn Soup 1(1)portion
Caramel Pudding 1(1) portion
Dumpling 1.5(1.5) portions
Corn Soup 1(1)portion
Seafood Spaghetti with Tomato Sauce 1(1) portion
Pork Bun 1(1) portion
Soy Milk 1(1) portion
Black Tea 1(1) portion
Whole Grains & Starches 14.5(14.5) servings
Fats & Nuts 9(9.3) servings
Meats & Proteins 10(9.65) servings
Sugar 72(72)g
Vegetables 0.5(0.5) serving
Fruits 0(0) serving
Low-Fat Milk 1(0.6) serving
Carbohydrate 1197(1196.8)kcal
Protein 407(407.4)kcal
Fat 874(874.35)kcal
FGB 1(0.66)
PCC 48(48.29)%
PCP 16(16.44)%
PCF 35(35.28)%
PCR 124(123.93)%
PCR: Percentage of Caloric Ratio
FGB: Food Group Balance
DHL: Dietary Healthy Level
DO: Desired Output
DHL_Method 3.4(3.42)
DHL_DO 4(4)
PCC: Percentage of Calories from Carbohydrate
PCP: Percentage of Calories from Protein
PCF: Percentage of Calories from Fat
R_class_Semantic
VeryLow
Low
Medium
High
VeryHigh
Recommended Semantics Layer

Personalized diet recommendation

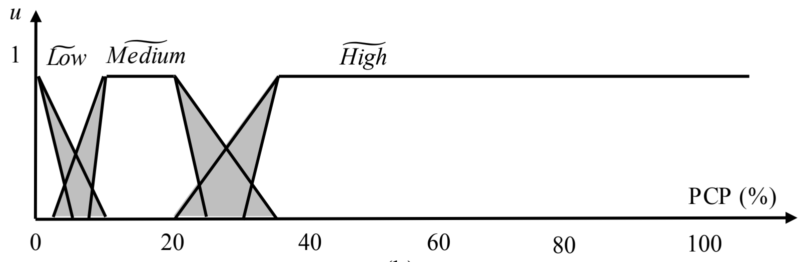


Step 6.2
Taiwan
FDA
Domain Experts
Step 1
Ontology Experts
Step 2
Subjects
Step 3
Step 8
Personalized KB
Food Item Database
Meal Record Database
Adaptive Personal Diet Assessment and Recommendation Ontology
T2GFML Repository
Step 5
Training Data
Step 6
Step 4
T2 FS-based Learning Mechanism
Step 6.1
T2 FS-based Genetic Learning Mechanism
Step 6.3
T2 FS-based Fuzzy Inference Mechanism
Step 6.4
Linguistic Knowledge Discovery Mechanism
Step 4.1
Nutritional Balanced Mechanism
Step 4.2
Caloric Balanced Mechanism
Step 4.3
Type-2 Six-Food-Group Balanced Mechanism
Balanced Computation Mechanism
Step 7.1
Dietary Health Level Repository
Step 7.2

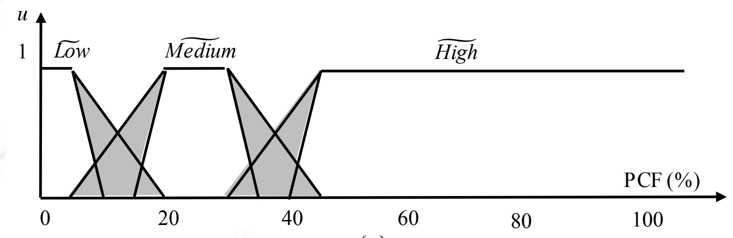
T2FS fuzzy variables



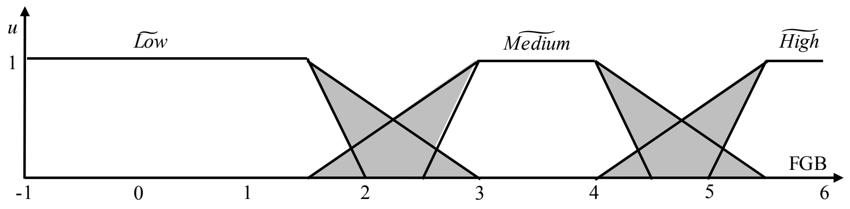
(a)



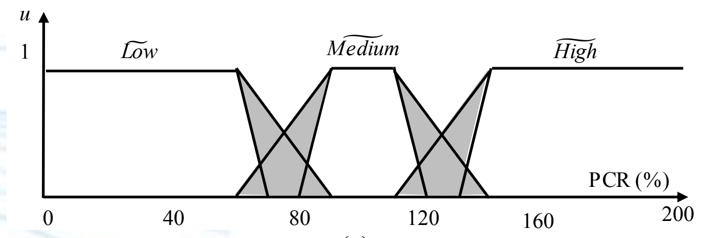
(b)



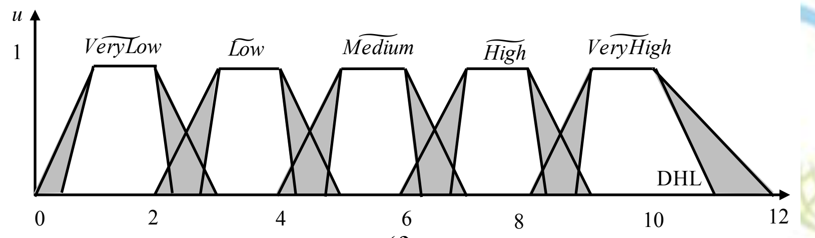
(c)



(d)

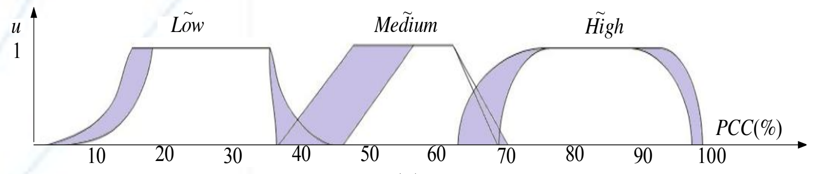


(e)

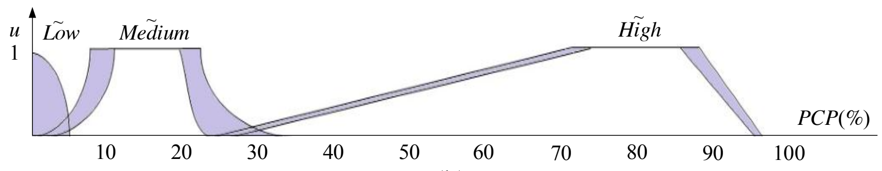


(f)

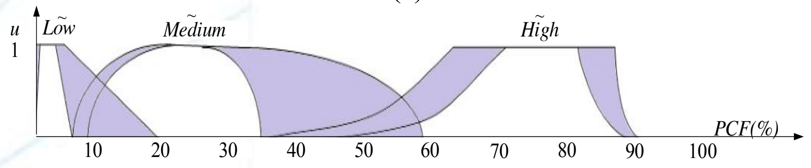
T2FS fuzzy variables



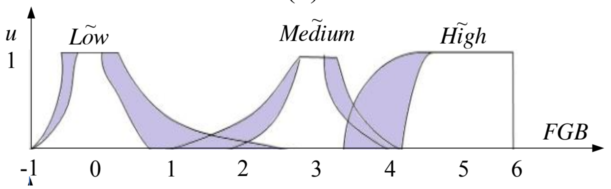
(a)



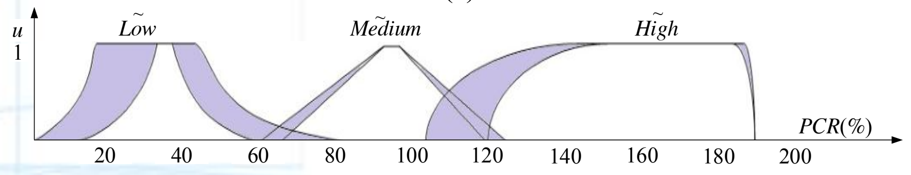
(b)



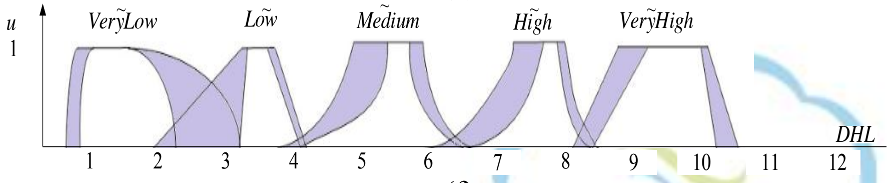
(c)



(d)



(e)

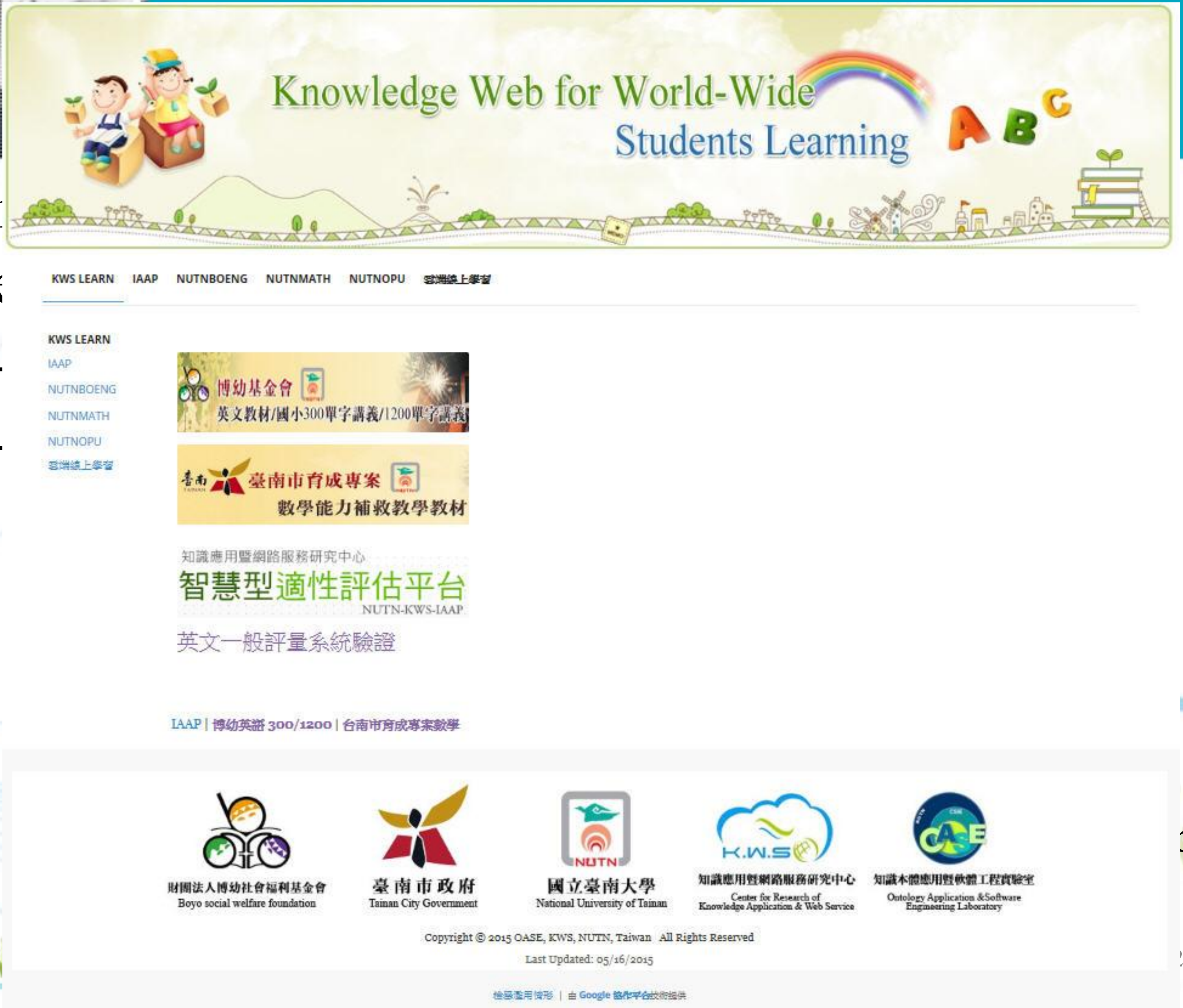


(f)

Adaptive Learning Application

- K
  Le
  –
  –

o.



KWS LEARN | IAAP | NUTNBOENG | NUTNMATH | NUTNOPU | 雲端線上學習

KWS LEARN
IAAP
NUTNBOENG
NUTNMATH
NUTNOPU
雲端線上學習

博幼基金會
英文教材/國小300單字講義/1200單字講義

臺南 TAINAN 臺南市育成專案
數學能力補救教學教材

知識應用暨網路服務研究中心
智慧型適性評估平台
NUTN-KWS-IAAP

英文一般評量系統驗證

IAAP | 博幼英語 300/1200 | 台南市育成專案數學

財團法人博幼社會福利基金會
Boyo social welfare foundation

臺南市政府
Tainan City Government

NUTN
國立臺南大學
National University of Tainan

K.W.S
知識應用暨網路服務研究中心
Center for Research of Knowledge Application & Web Service

CASE
知識本體應用暨軟體工程實驗室
Ontology Application &Software Engineering Laboratory

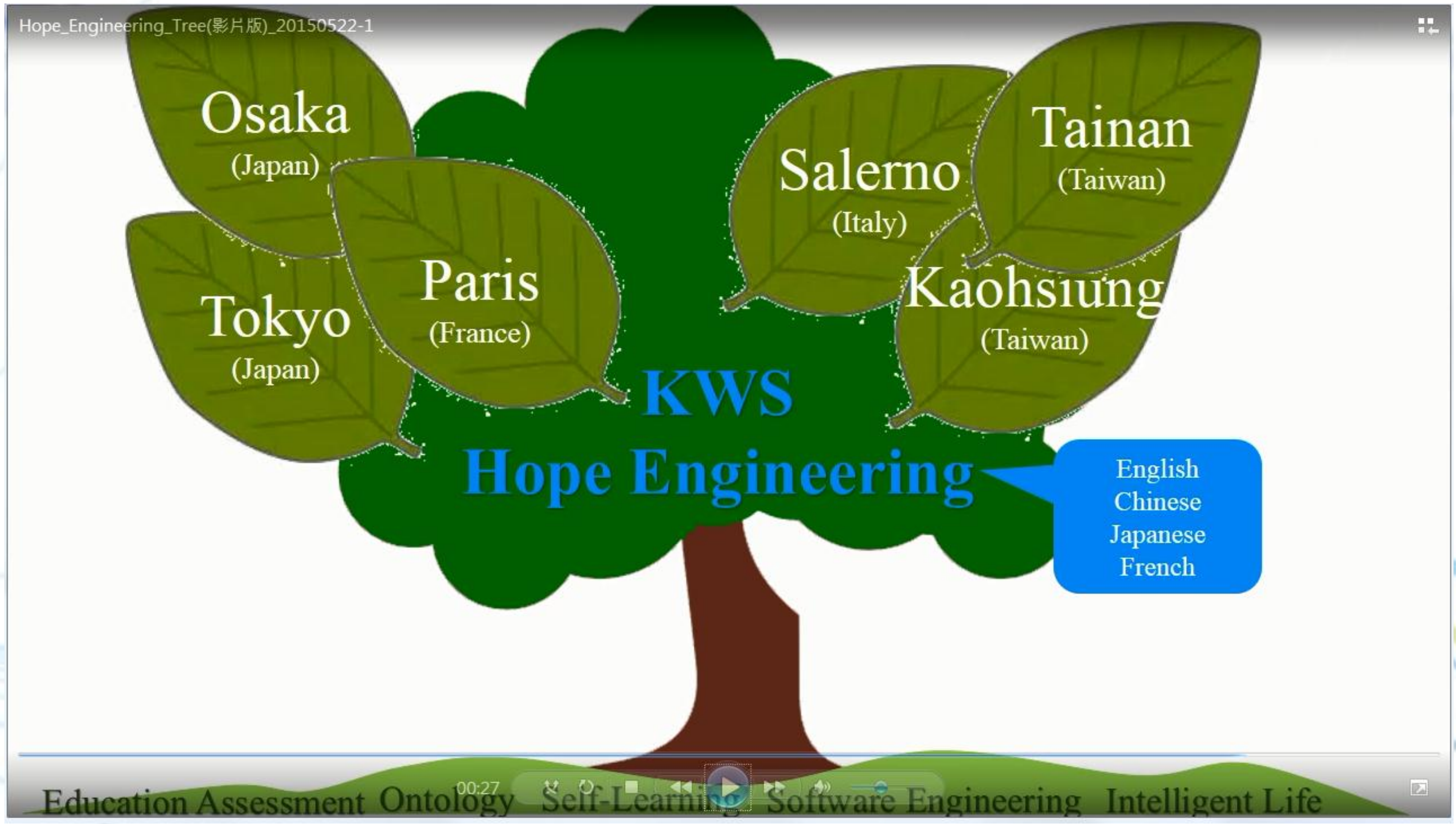
Copyright © 2015 OASE, KWS, NUTN, Taiwan All Rights Reserved
Last Updated: 05/16/2015

檢舉濫用情形 | 由 Google 協作平台技術提供

NUTN

KWS Hope Engineering

- KWS Hope Engineering


Hope_Engineering_Tree(影片版)_20150522-1
Osaka (Japan)
Tokyo (Japan)
Paris (France)
Salerno (Italy)
Tainan (Taiwan)
Kaohsiung (Taiwan)
KWS Hope Engineering
English Chinese Japanese French
Education Assessment Ontology Self-Learning Software Engineering Intelligent Life
00:27

English Teaching Materials (1/3)

- Boyo Social Welfare Foundation
  - 300 English words for elementary-school students
  - 1200 English words for elementary-school and junior-high-school students

IEEE World Congress on Computational Intelligence
24-29 July 2016, Vancouver, Canada
IEEE WCCI 2016
Vancouver Canada

FUZZ-IEEE 2016 Tutorials: FUZZ-IEEE-03 Part 2

Fuzzy Markup Language

Giovanni Acampora

Nottingham Trent University, Nottingham, UK

Fuzzy Markup Language Toward the first standard in Computational Intelligence

Technical Committee on Standards

Task Force on Novel Standard Proposal

Giovanni Acampora, Plamen Angelov and Bruno Di Stefano

December 11th, 2011

IEEE Computational Intelligence Society

MIMICKING NATURE FOR PROBLEM SOLVING

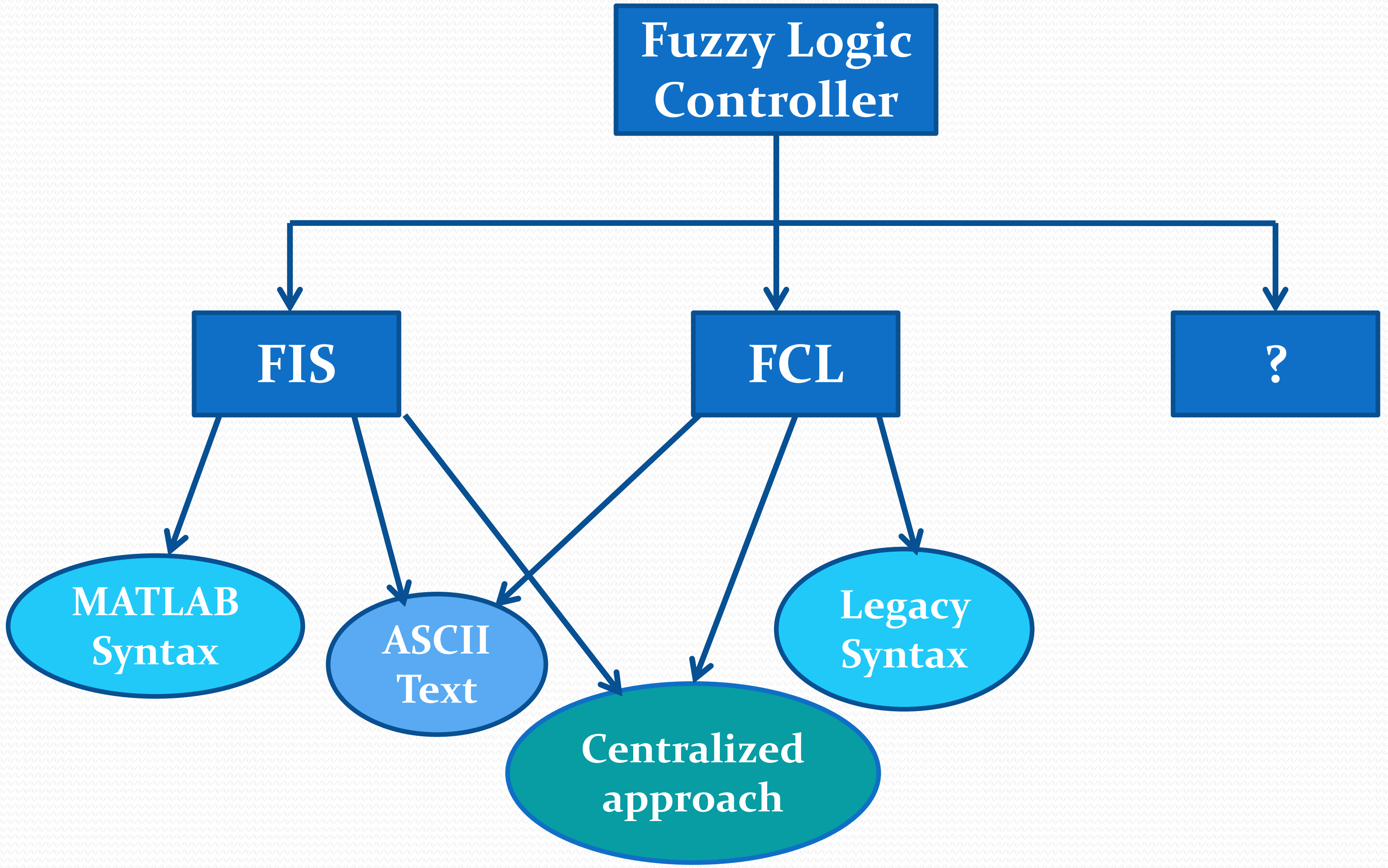
Goal

Propose the Fuzzy Markup Language (FML)
as a standard tool for the design and implementation
of Fuzzy Systems

Motivations

- A standard tool for fuzzy logic is necessary for:
  - Designing fuzzy controllers in hardware independent way;
  - Distributing fuzzy systems in complex pervasive environment;
  - Representing fuzzy rules in a unified way in order to allow different scientist groups to compare the performance of their learning algorithms;
  - Allowing conference organizers to use a well-defined approach for organizing fuzzy based competitions.

Current Tools for FC Design



Fuzzy Control Language (FLC)

- FCL was standardized by IEC 61131-7

  INTERNATIONAL ELECTROTECHNICAL COMMISSION (IEC)

  TECHNICAL COMMITTEE No. 65: INDUSTRIAL PROCESS MEASUREMENT AND CONTROL

  SUB-COMMITTEE 65 B: DEVICES

- It is a domain-specific programming language
  - it has no features unrelated to fuzzy logic;
  - one does not write a program in FCL, but one may write part of it in FCL.

FCL drawbacks

- FCL is **static**
  - it contains information about the data but not about the functions needed to process the data;
  - It cannot generate an executable program;
- FCL does not support “hedges”;
- FCL lacks support for higher-order fuzzy sets;
- FCL does **not allow binding data and function,** a standard feature of OOP languages;

FCL drawbacks

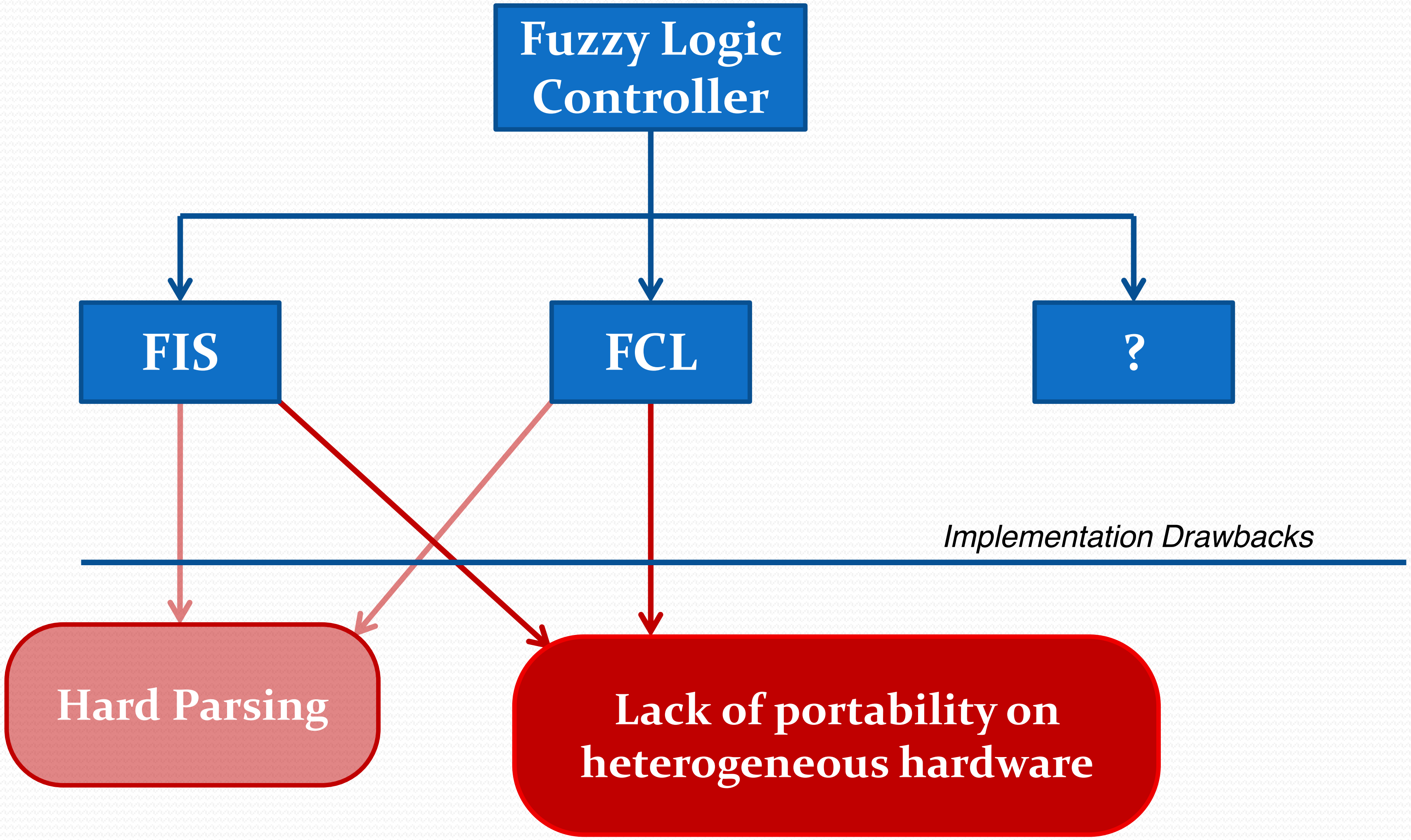
- An FCL description of an algorithm may result in different implementations of the algorithm;

- FCL was definitely an accomplishment in the 90s because it allowed practitioners no exchange information about fuzzy algorithms;

- However, it reflected the close world of proprietary systems, where interchange of building blocks was discouraged in the attempt to lock clients into corporate platforms.

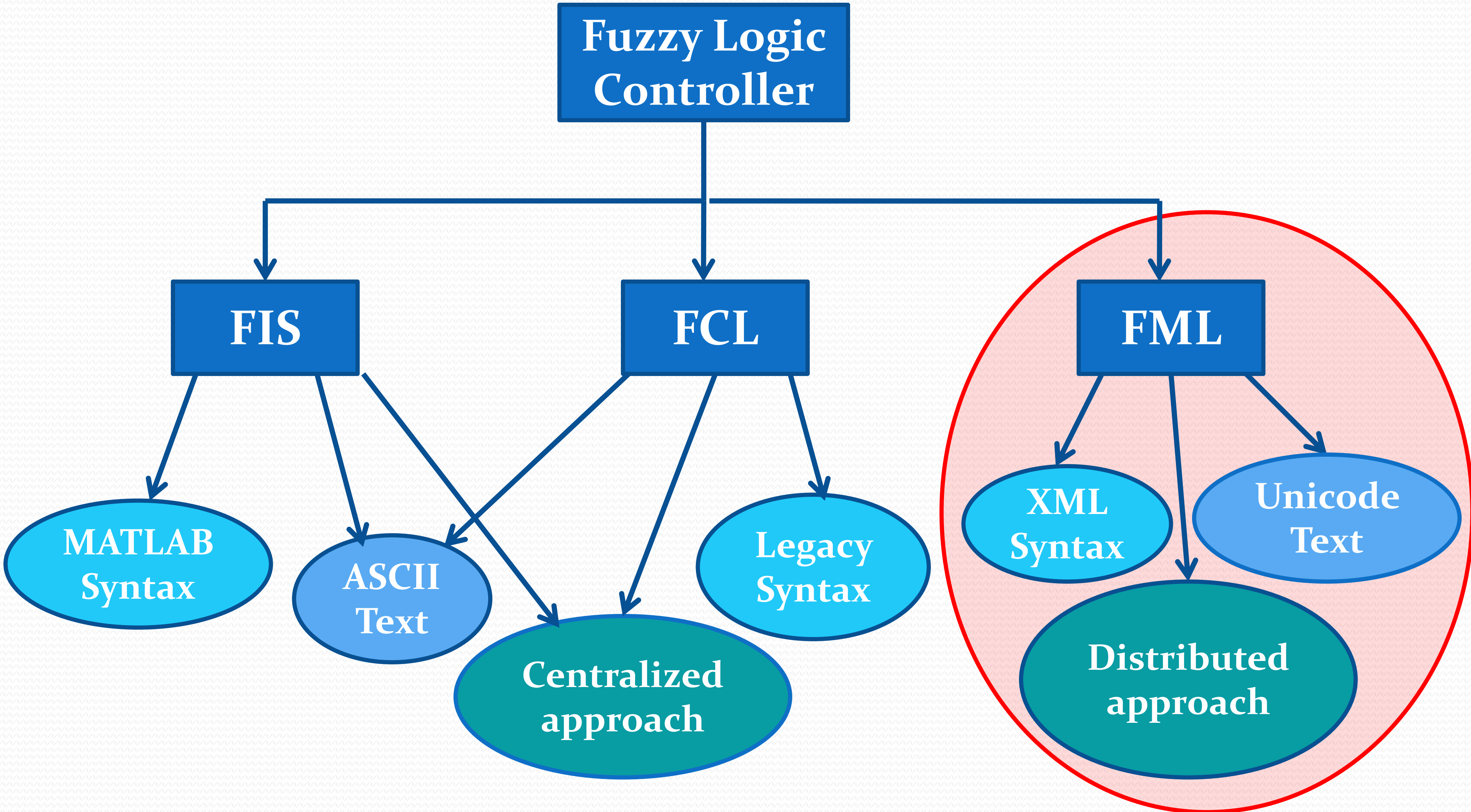
Current Tools for FC Design

Drawbacks



A New Paradigm for Fuzzy Control Design

Fuzzy Markup Language



FML Idea

- FML is a XML-based language which allows FC designers to model the controllers in a human-readable and hardware independent way in highly distributed environments.

↓

FML allows to model Transparent Fuzzy Controllers

FML Benefits

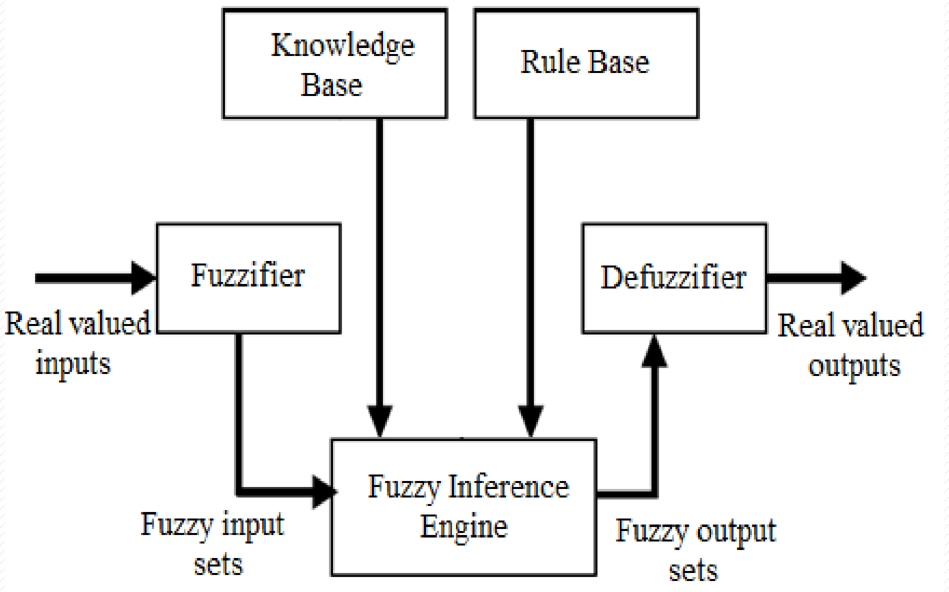
- Minimize effort to reprogram controllers on different hardware architecture.
- Enable the distribution of fuzzy inference engine task in order to optimize performances and support the development of application based on the sensors network paradigm.

… hereafter

- The Fuzzy Markup Language: Theory and Practice
- FML Visual Environment for Transparent Fuzzy Systems Design
- FML Applications

Fuzzy Markup Language

Fuzzy Logic Controller Structure



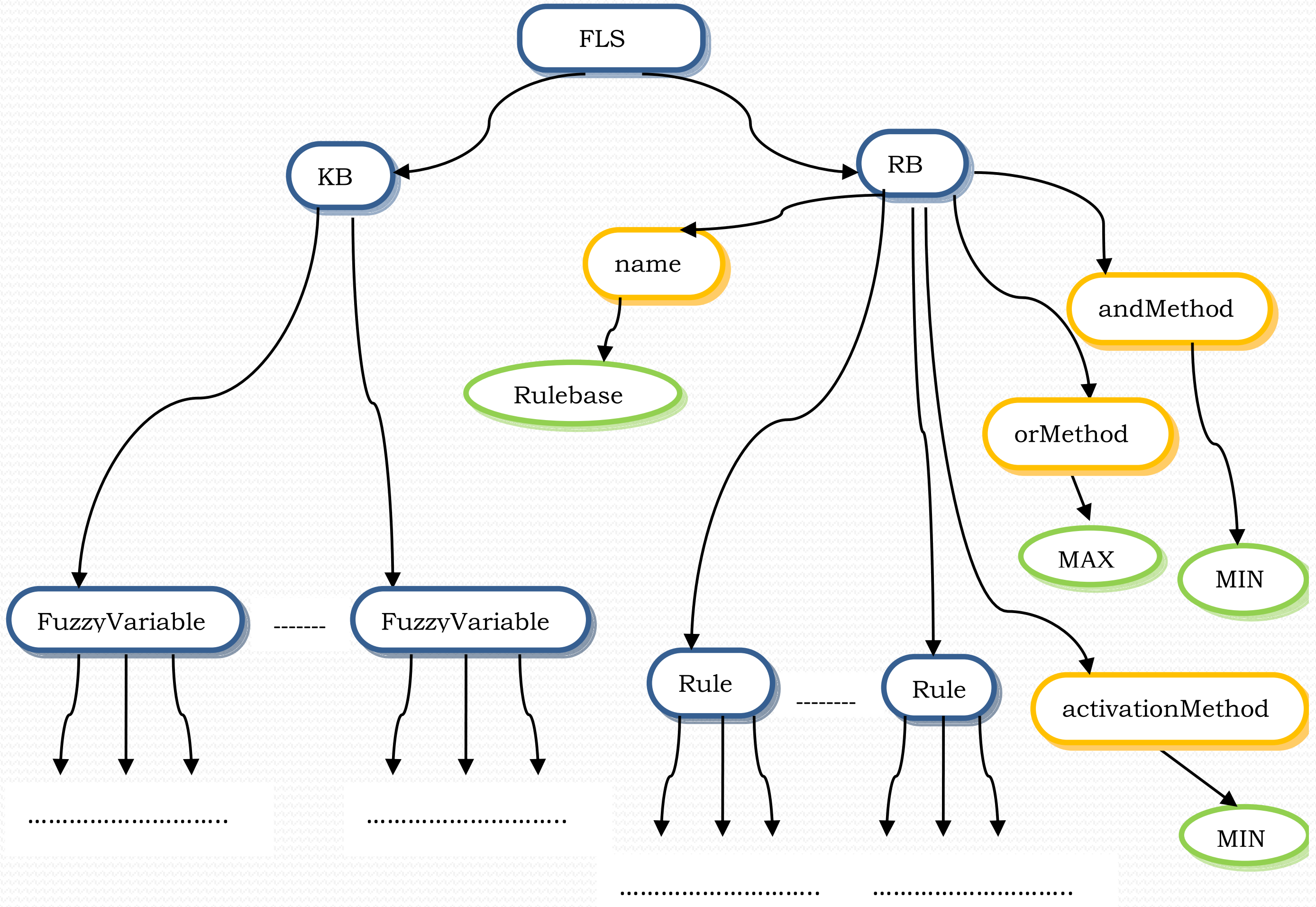
- Simpler than PID controllers (based on differential equation)
- Use a linguistic approach (*If…Then*) to define the devices behavior

The novel vision of a FLC

- An alternative vision of FLCs implementation is necessary to model the controllers in hardware independent way.

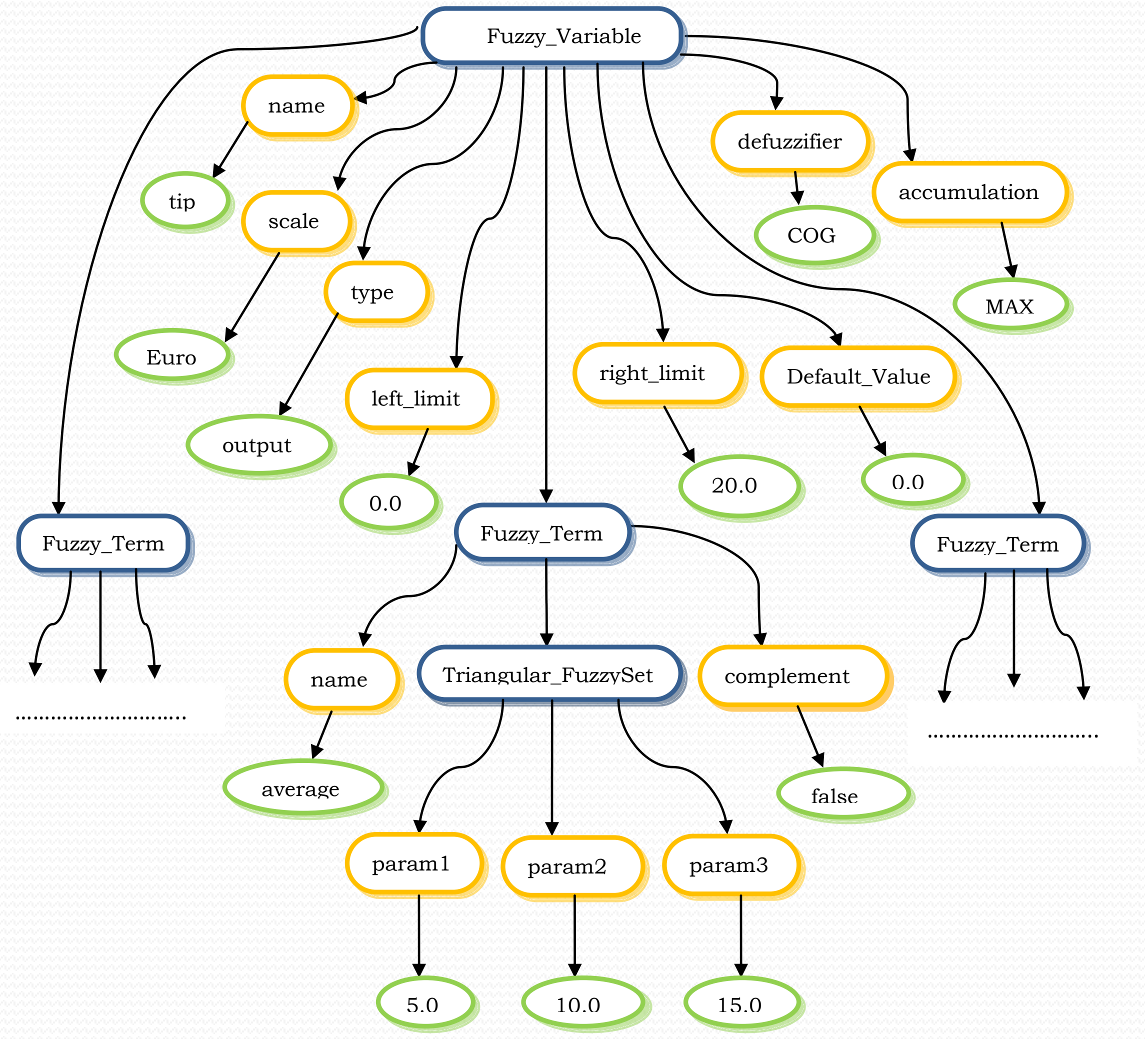
- This novel vision is based on the labeled tree idea, a data structure defined by means of the well-known graph theory.
  - A labeled tree is a connected and acyclic labeled graph, i.e, a graph where each node is associated with a label.

FLC Labeled Tree



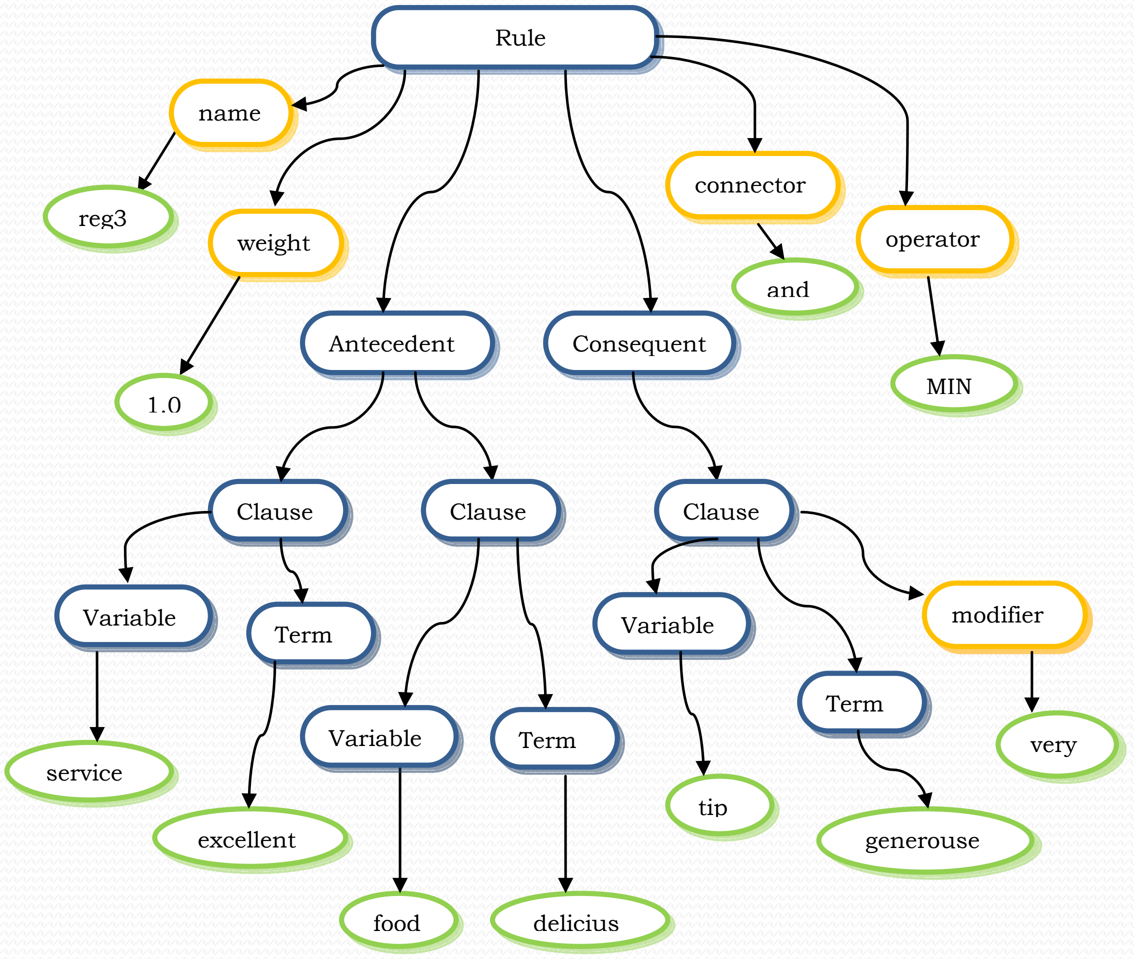
FLC Labeled Tree

Labeled tree for the variable tip



FLC Labeled Tree

Labeled Tree for rule 3



Labeled Tree & XML

- The labeled trees are data models derived by the XML-based document representation.
- So, if a FLC can be modeled through a labeled tree then it is representable by means of a corresponding XML document.

- XML is the main technology for abstraction data:
  - The XML representation of FLC allows designers to model the controller in a human-readable and hardware independent way.
  - FML is the XML-based language modeling FLC.

From labeled tree to FML

- FML is the XML-based language modeling FLCs, i.e, a collection of tags and attributes that are individuated starting from the analysis of the FLC labeled tree.

- So, thanks to FML, it is possible to implement the same FLC on different hardware without additional design and implementation steps.

$\Downarrow$

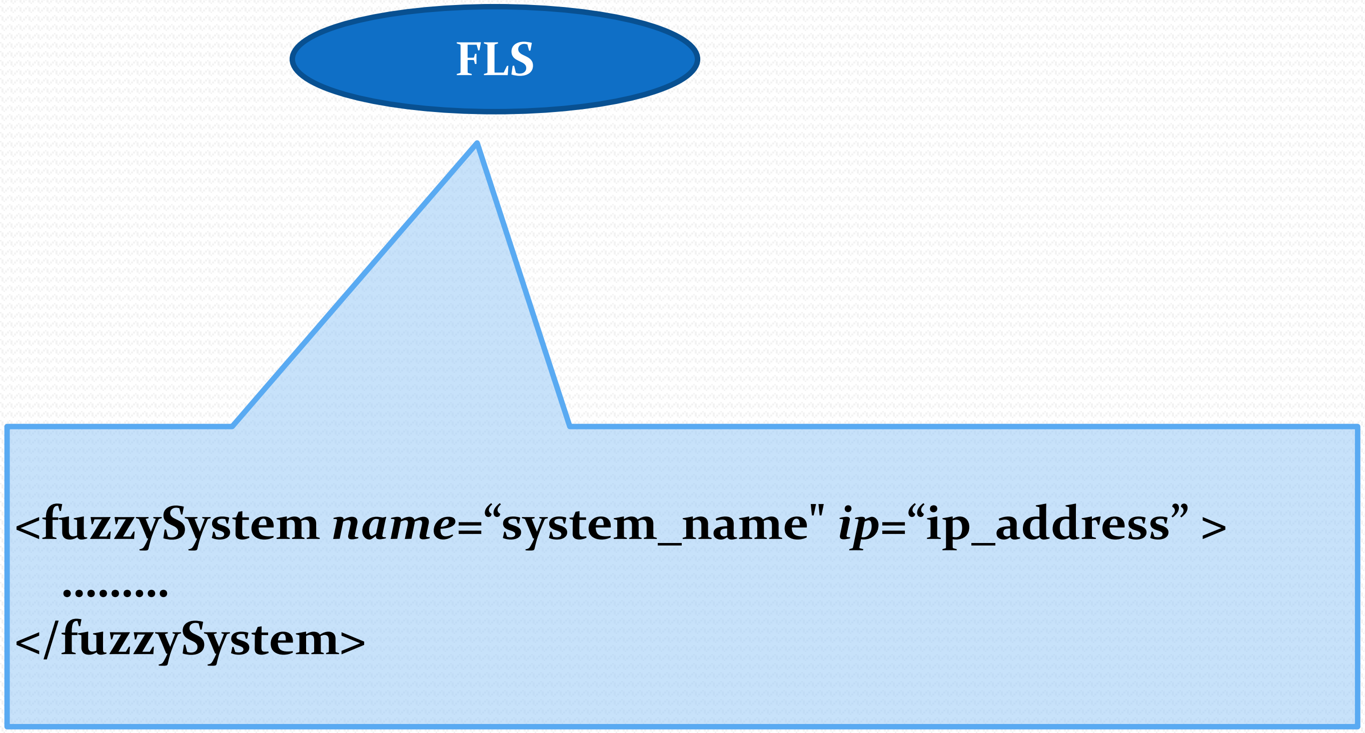
Transparent Fuzzy Control

FML definition

- FML is essentially composed by three layers:
  - **XML** in order to create a new markup language for fuzzy logic control;
  - a document type definition (DTD), initially, and now a **XML Schema** in order to define the legal building blocks;
  - Extensible Stylesheet Language Transformations (**XSLT**) in order to convert a fuzzy controller description into a specific programming language.

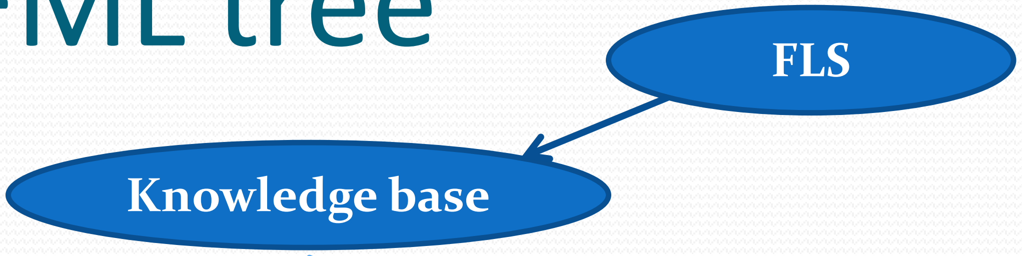
- FML permits to model the two well-known fuzzy controllers: Mamdani and TSK.

FML tree



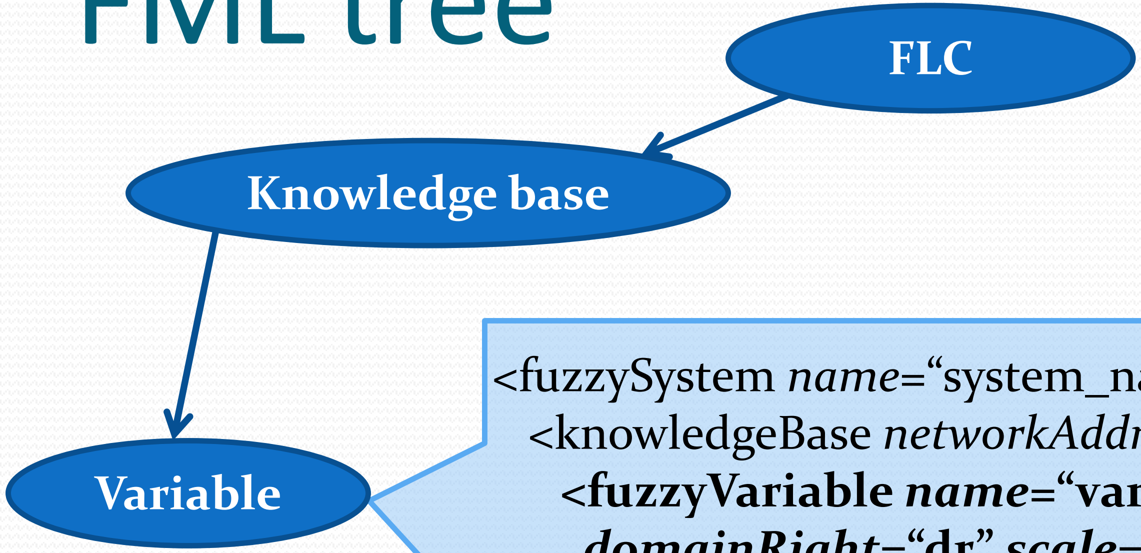
```
<fuzzySystem name="system_name" ip="ip_address" >
   .........
</fuzzySystem>
```

FML tree



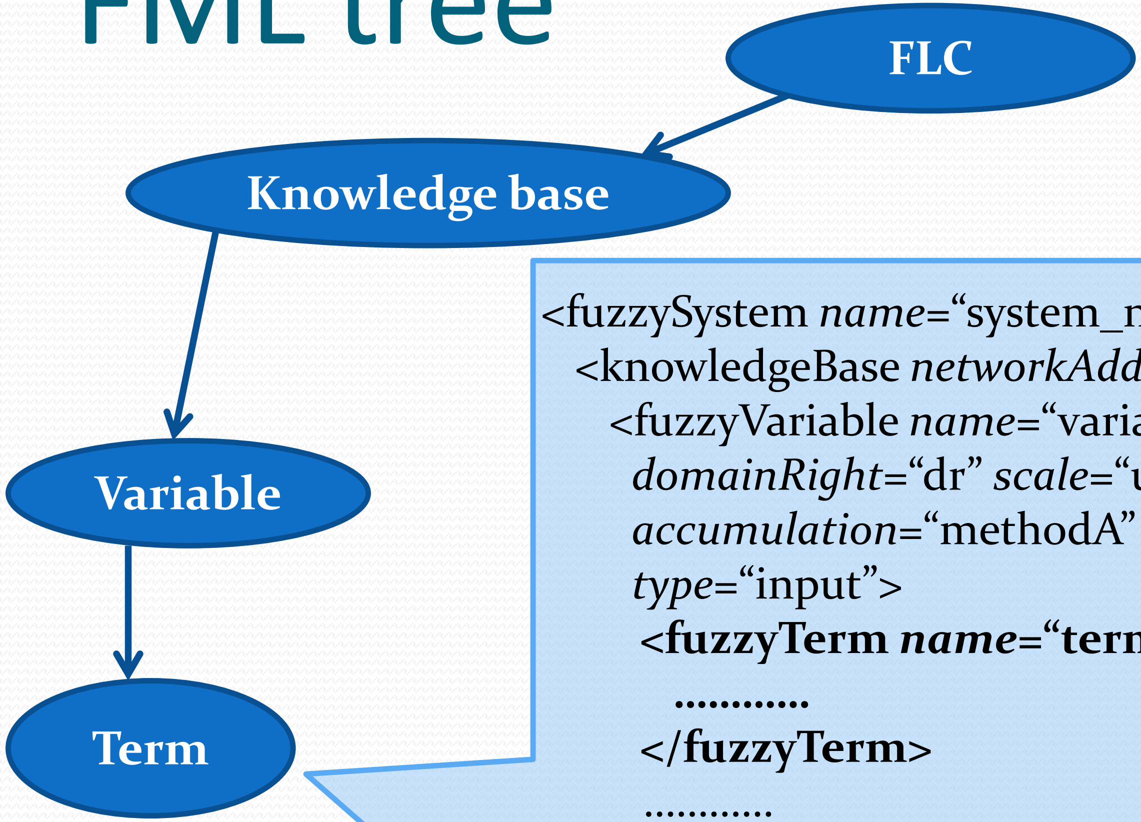
```
<fuzzySystem name="system_name" networkAddress="ip_address" >
  <knowledgeBase ip="ip_address" >
  ........
  </knowledgeBase>
  ........
</fuzzySystem>
```

FML tree



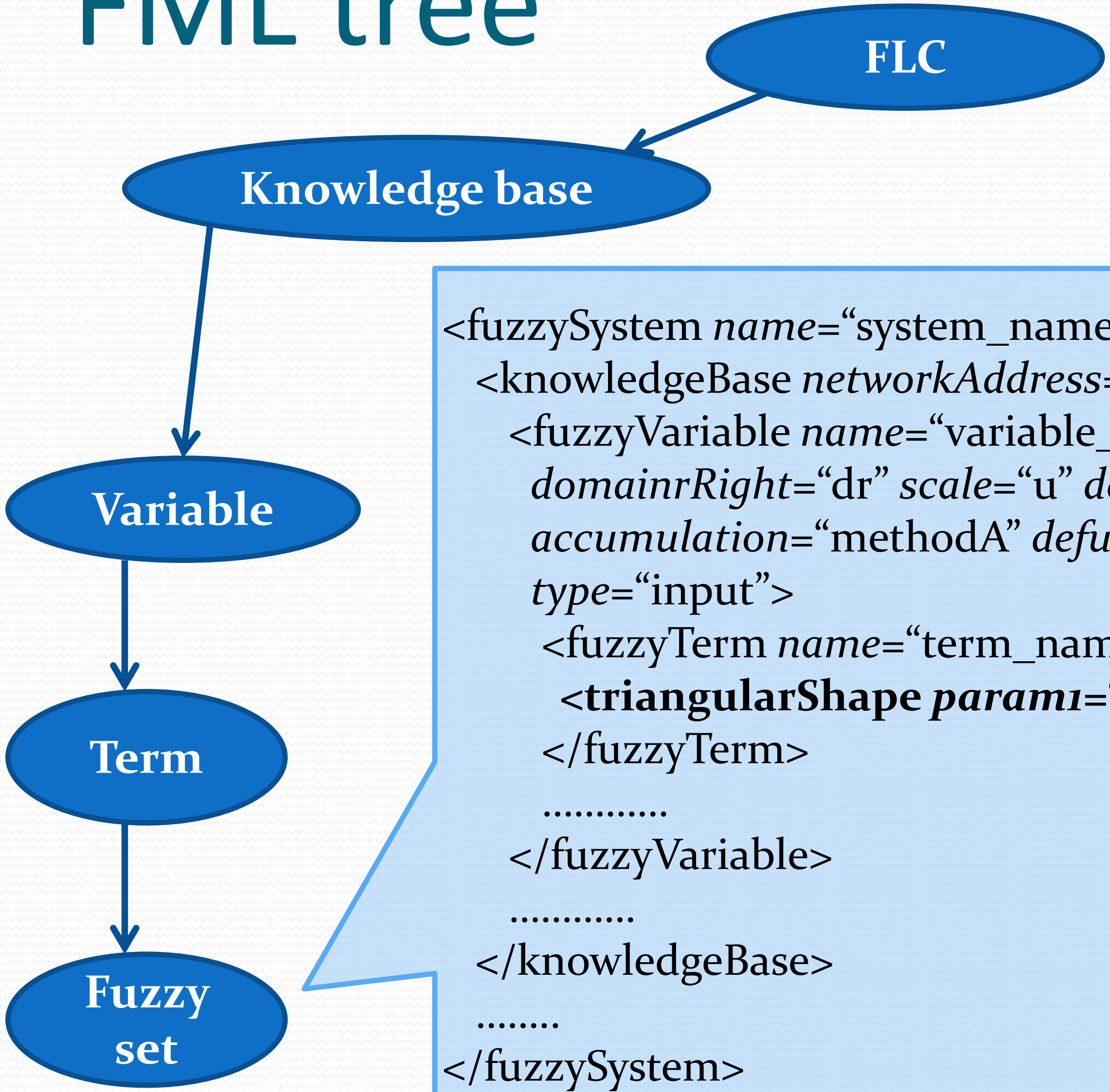
```
<fuzzySystem name="system_name" networkAddress="ip_address">
  <knowledgeBase networkAddress="ip_address" >
    <fuzzyVariable name="variable_name" domainLeft="dl"
      domainRight="dr" scale="u" defaultValue="dv"
      accumulation="methodA" defuzzifier="methodD"
      type="output">
        ......
    </fuzzyVariable>
    ........
  </knowledgeBase>
  ........
</fuzzySystem>
```

FML tree



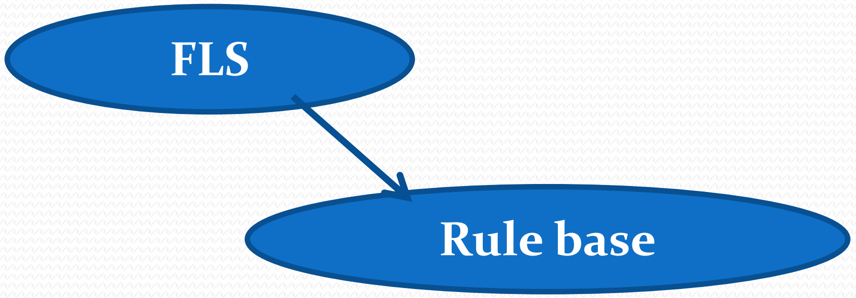
```
<fuzzySystem name=“system_name" networkAddress=“ip_address”>
  <knowledgeBase networkAddress=“ip_address” >
    <fuzzyVariable name=“variable_name” domainLeft=“dl”
      domainRight=“dr” scale=“u” defaultValue=“dv”
      accumulation=“methodA” defuzzifier=“methodD”
      type=“input”>
       <fuzzyTerm name=“term_name” complement=“false”>
         ............
       </fuzzyTerm>
       ............
    </fuzzyVariable>
    ............
  </knowledgeBase>
  ........
</fuzzySystem>
```

FML tree



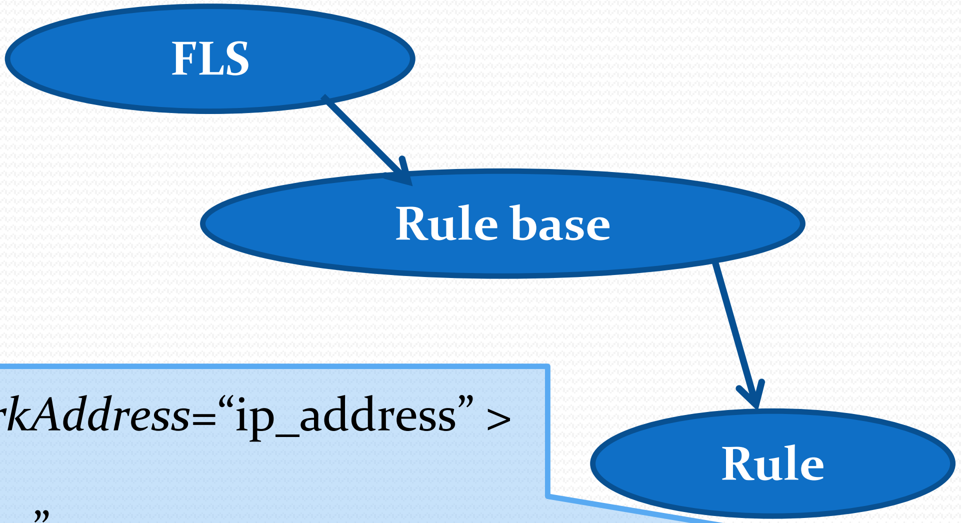
```
<fuzzySystem name="system_name" networkAddress="ip_address" >
  <knowledgeBase networkAddress="ip_address" >
    <fuzzyVariable name="variable_name" domainLeft="dl"
     domainrRight="dr" scale="u" defaultValue="dv"
     accumulation="methodA" defuzzifier="methodD"
     type="input">
      <fuzzyTerm name="term_name" complement="false">
       <triangularShape param1="p1" param2="p2" param3="p3"/>
      </fuzzyTerm>
      ............
    </fuzzyVariable>
    ............
  </knowledgeBase>
  ........
</fuzzySystem>
```

FML tree



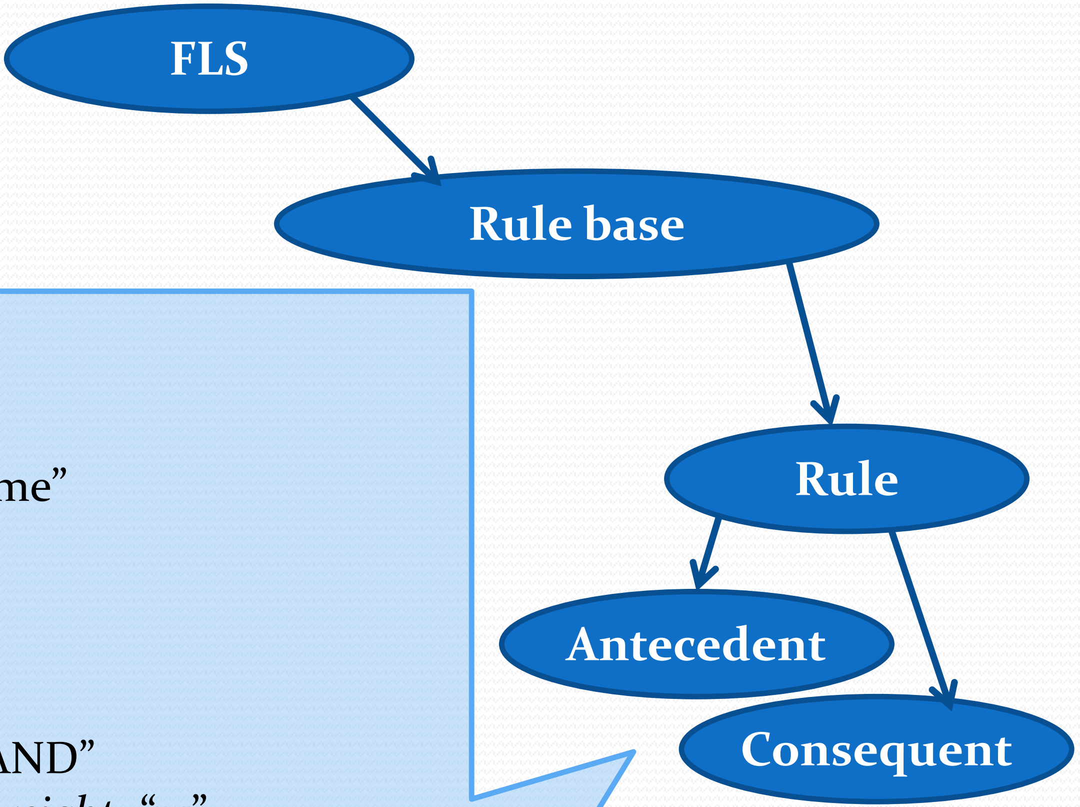
```
<fuzzySystem name="system_name" networkAddress="ip_address" >
  ........
  <mamdaniRuleBase name="rulebase_name"
andMethod="And_Meth"
     orMethod="Or_Meth" activationMethod="Activation_Meth"
    networkAddress="ip_address" >
     ........
  </mamdaniRuleBase>
</fuzzySystem>
```

FML tree



```
<fuzzySystem name="system_name" networkAddress="ip_address" >
  ........
  <mamdaniRuleBase name="rulebase_name"
andMethod="And_Meth"
     orMethod="Or_Meth" activationMethod="Activation_Meth"
networkAddress="ip_address" >
     <rule name="rule_name" connector="AND"
                    andMethod="And_Meth" weight="w">
        ........
     </rule>
     ........
  </mamdaniRuleBase>
</fuzzySystem>
```

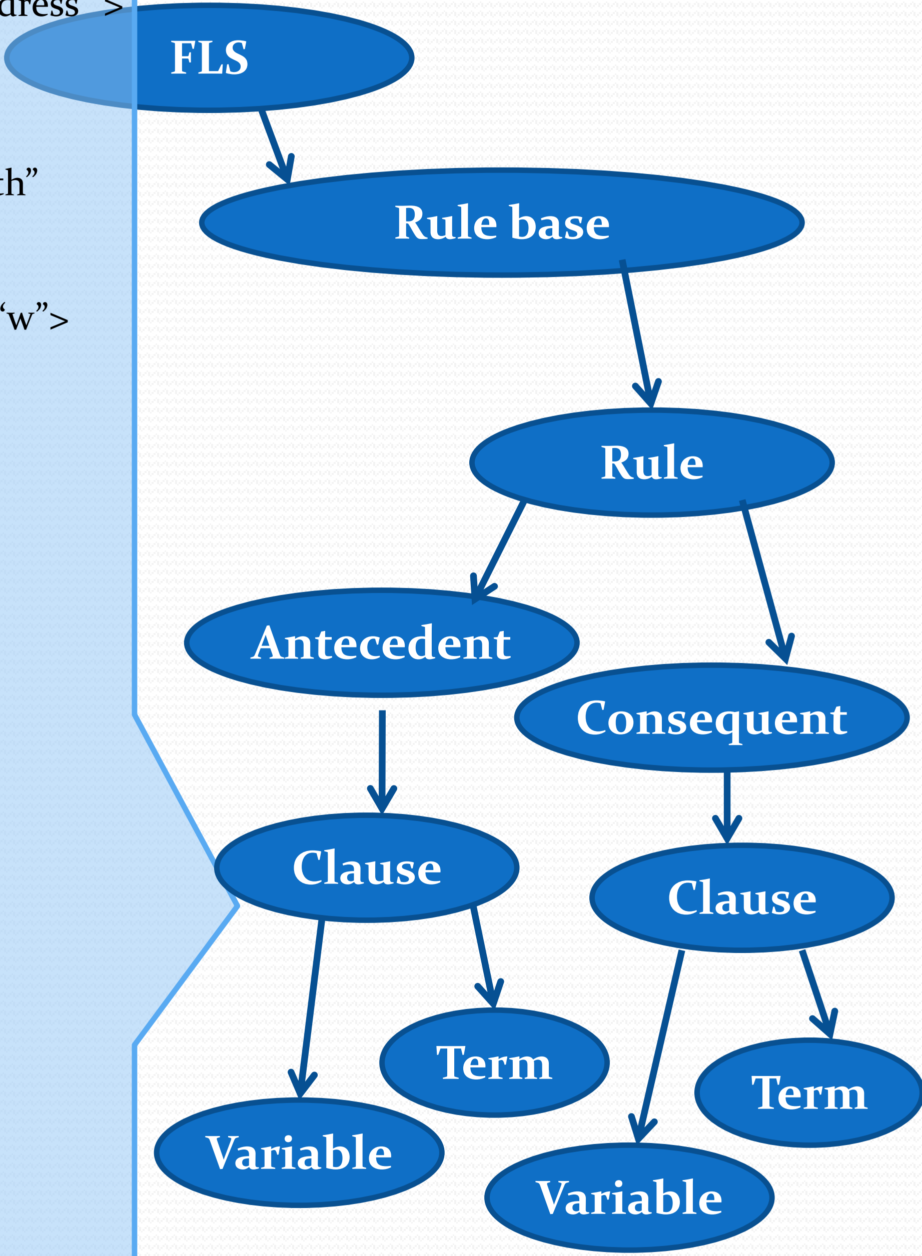
FML tree



```
<fuzzySystem name="system_name"
networkAddress="ip_address" >
   ........
   <mamdaniRuleBase name="rulebase_name"
      andMethod="And_Meth"
      orMethod="Or_Meth"
      activationMethod="Activation_Meth"
      networkAddress="ip_address" >
       <rule name="rule_name" connector="AND"
                      andMethod="And_Meth" weight="w">
          <antecedent>......</antecedent>
          <consequent>......</consequent>
       </rule>
       ........
   </mamdaniRuleBase>
</fuzzySystem>
```

FML tree

```
<fuzzySystem name="system_name" networkAddress="ip_address" >
  ........
  <mamdaniRuleBase name="rulebase_name"
andMethod="And_Meth""
      orMethod="Or_Meth" activationMethod="Activation_Meth"
      netwrokAddress="ip_address" >
      <rule name="rule_name" connector="AND/OR"
                              andMethod="And_Method" weight="w">
        <antecedent>
          <clause>
             <variable>Variable_Name</variable>
             <term>Term_Name</term>
          </clause>
          ........
        </antecedent>
        <consequent>
           <then>
                <clause>
                   <variable>Variable_Name</variable>
                   <term>Term_Name</term>
                </clause>
             </then>
          ........
        </consequent>
     </rule>
     ........
  </mamdaniRuleBase>
</fuzzySystem>
```



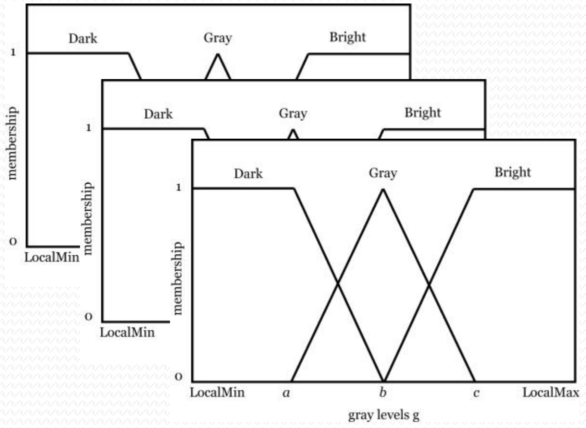
FML grammar

- The labeled tree' s labels and relationships have to be represented by means of a grammar in order to be used in a computing scenario.
- This grammar definition can be accomplished by means of XML tools able to translate the FLC labeled tree description in a context free grammar.

- Latest version of FML grammar has been developed through **XML Schema**

FML variable grammar

```
<xs:complexType name="fuzzyVariableType">
      <xs:sequence>
         <xs:element name="fuzzyTerm" type="fuzzyTermType" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Fuzzy Set</xs:documentation>
            </xs:annotation>
         </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:ID" use="required"/>
      <xs:attribute name="scale" type="xs:string" />
      <xs:attribute name="domainleft" type="xs:float"  use="required"/>
      <xs:attribute name="domainright" type="xs:float" use="required"/>
      <xs:attribute name="type" type="typeType" default="input"/>
      <xs:attribute name="accumulation" type="accumulationType" default="MAX" />
      <xs:attribute name="defuzzifier" type="defuzzifierType" default="COG" />
      <xs:attribute name="defaultValue" type="xs:float" default="0"/>
      <xs:attribute name="networkAddress" default="127.0.0.1" type="networkAddressType"/>
   </xs:complexType>
   <xs:complexType name="fuzzyTermType">
      <xs:choice>
         <xs:element name="rightLinearShape" type="twoParamType" >
            <xs:annotation>
               <xs:documentation>Right Linear Fuzzy Set</xs:documentation>
            </xs:annotation>
          </xs:element>
          ........
      </xs:choice>
      <xs:attribute name="name" type="xs:ID" use="required"/>
      ............
   </xs:complexType>
```



Example: tipper.fml

```
<?xml version="1.0" encoding="UTF-8"?>
<fuzzySystem name="newSystem" networkAddress="127.0.0.1">
  <knowledgeBase>
    <fuzzyVariable name="food" domainLeft="0.0" domainRight="10.0" scale="" type="input">
      <fuzzyTerm name="delicius" complement="false">
        <trapezoidShape param1="7.0" param2="9" param3="10.0" param4="10.0"/>
      </fuzzyTerm>
      <fuzzyTerm name="rancid" complement="false">
        <trapezoidShape param1="0.0" param2="0.0" param3="1" param4="3"/>
      </fuzzyTerm>
    </fuzzyVariable>
    <fuzzyVariable name="service" domainLeft="0.0" domainRight="10.0" scale="" type="input">
      <fuzzyTerm name="excellent" complement="false">
        <leftGaussianShape param1="10.0" param2="1.5"/>
      </fuzzyTerm>
      <fuzzyTerm name="good" complement="false">
        <PIShape param1="5.0" param2="3.0"/>
      </fuzzyTerm>
      <fuzzyTerm name="poor" complement="false">
        <rightGaussianShape param1="0.0" param2="1.5"/>
      </fuzzyTerm>
    </fuzzyVariable>
```

Example : tipper.fml

```
<fuzzyVariable name="tip" domainLeft="0.0" domainRight="20.0" scale="null"
   defaultValue="0.0" accumulation="MAX" defuzzifier="COG" type="output">
        <fuzzyTerm name="average" complement="false">
           <triangularShape param1="5.0" param2="10.0" param3="15.0"/>
        </fuzzyTerm>
        <fuzzyTerm name="cheap" complement="false">
           <triangularShape param1="0.0" param2="5.0" param3="10.0"/>
        </fuzzyTerm>
        <fuzzyTerm name="generous" complement="false">
           <triangularShape param1="10.0" param2="15.0" param3="20.0"/>
        </fuzzyTerm>
      </fuzzyVariable>
   </knowledgeBase>
```

Example : tipper.fml

```
<mamdaniRuleBase name="Rulebase1" andMethod="MIN"
    orMethod="MAX" activationMethod="MIN" >
     <rule name="reg1" connector="or"
                orMethod="MAX" weight="1.0">
       <antecedent>
         <clause>
           <variable>food</variable>
           <term>rancid</term>
         </clause>
         <clause>
           <variable>service</variable>
           <term>poor</term>
         </clause>
       </antecedent>
       <consequent>
        <then>
         <clause>
           <variable>tip</variable>
           <term>cheap</term>
         </clause>
        </then>
       </consequent>
     </rule>
```

```
<rule name="reg2" connector="or"
           andMethod="MAX" weight="1.0">
     <antecedent>
       <clause>
         <variable>service</variable>
         <term>good</term>
       </clause>
      </antecedent>
      <consequent>
       <then>
        <clause>
         <variable>tip</variable>
         <term>average</term>
        </clause>
       </then>
      </consequent>
    </rule>
```

Example : tipper.fml

```
<rule name="reg3" connector="or"  andMethod="MAX“ weight="1.0">
      <antecedent>
        <clause>
          <variable>service</variable>
          <term>excellent</term>
        </clause>
        <clause>
          <variable>food</variable>
          <term>delicius</term>
        </clause>
      </antecedent>
      <consequent>
        <then>
         <clause>
          <variable>tip</variable>
          <term>generouse</term>
         </clause>
        </then>
      </consequent>
    </rule>
  </mamdaniRuleBase>
</fuzzySystem>
```

FML Sinthesys

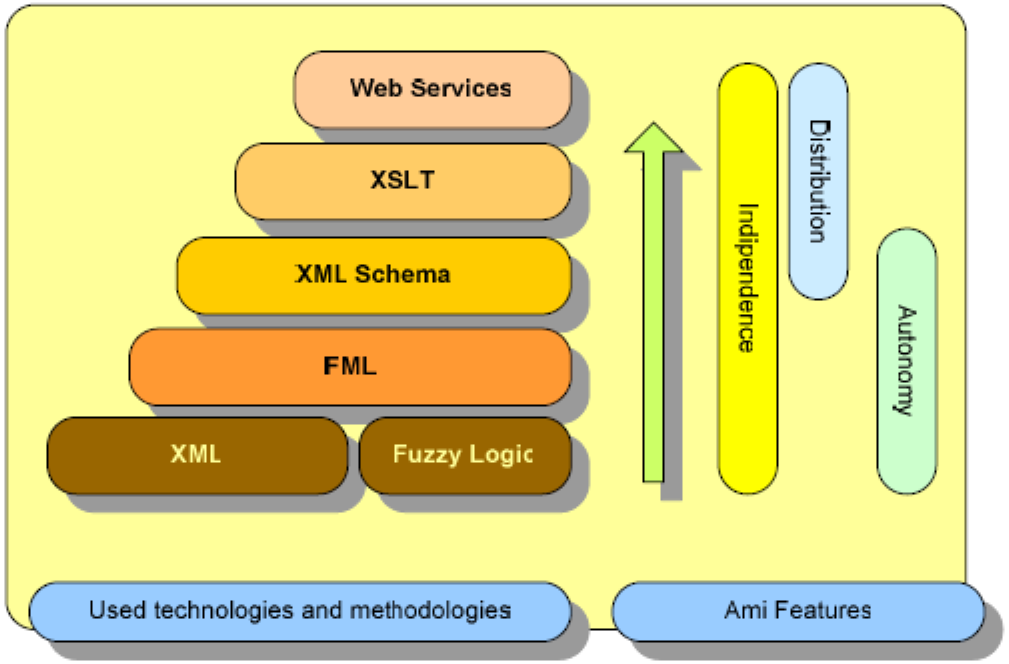
- FML represents a static and human oriented view of FLCs
- It is necessary to ‘compile’ FML programs.

- Different approaches has been used to compile FML programs
  - XSLT + Web Services
  - JAXB + Berkeley Sockets

XSLT

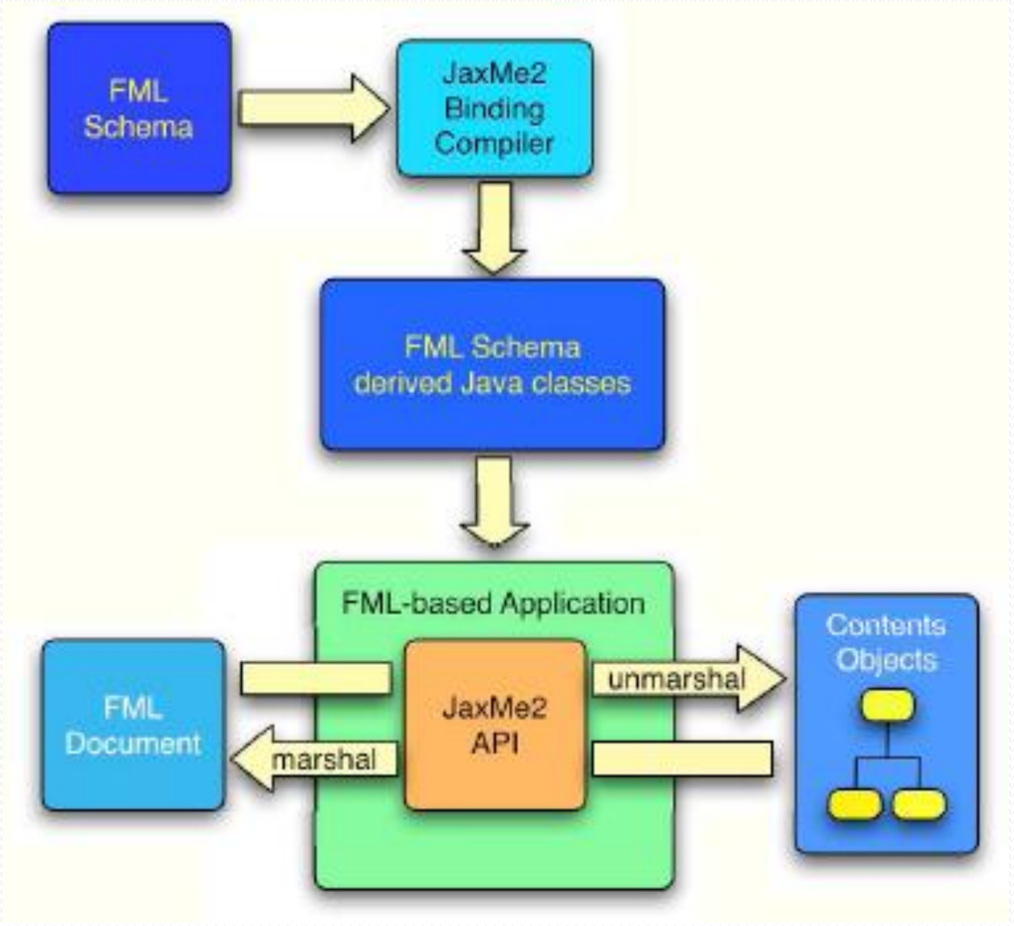
- XSLT languages translator is used to convert FML fuzzy controller in a general purpose computer language using an XSL file containing the translation description.

- It is possible to translate the FML programs into Java programs embedded in web services components in order to realize distribution features and to strong the hardware independence concept.



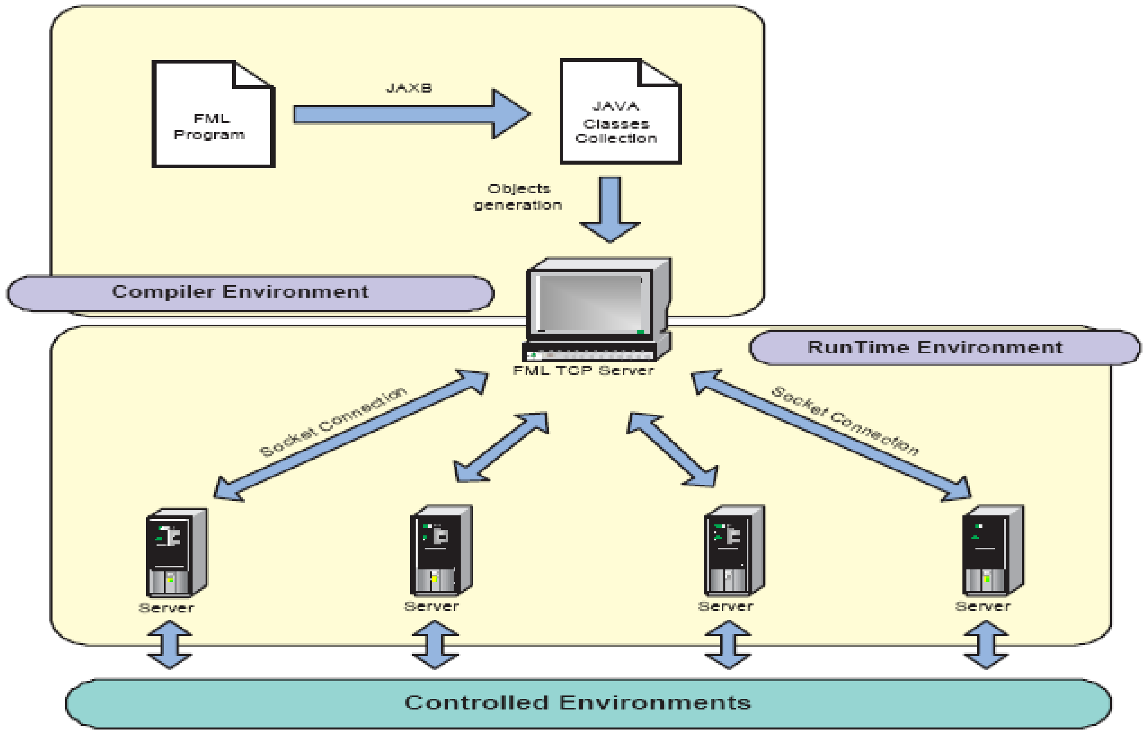
JAXB

- The JAXB XML Binding technology (or its open source version: JaxMe2) generates a Java classes hierarchy starting from the FML control description.



FML + JAXB + TCP/IP

- It is possible to integrate JAXB XML Binding technology with a TCP/IP Client/Server application to separate the real control from the controlled devices and, consequently, to obtain the total independence of the devices from the language used to code the fuzzy controller.



Distributing FML

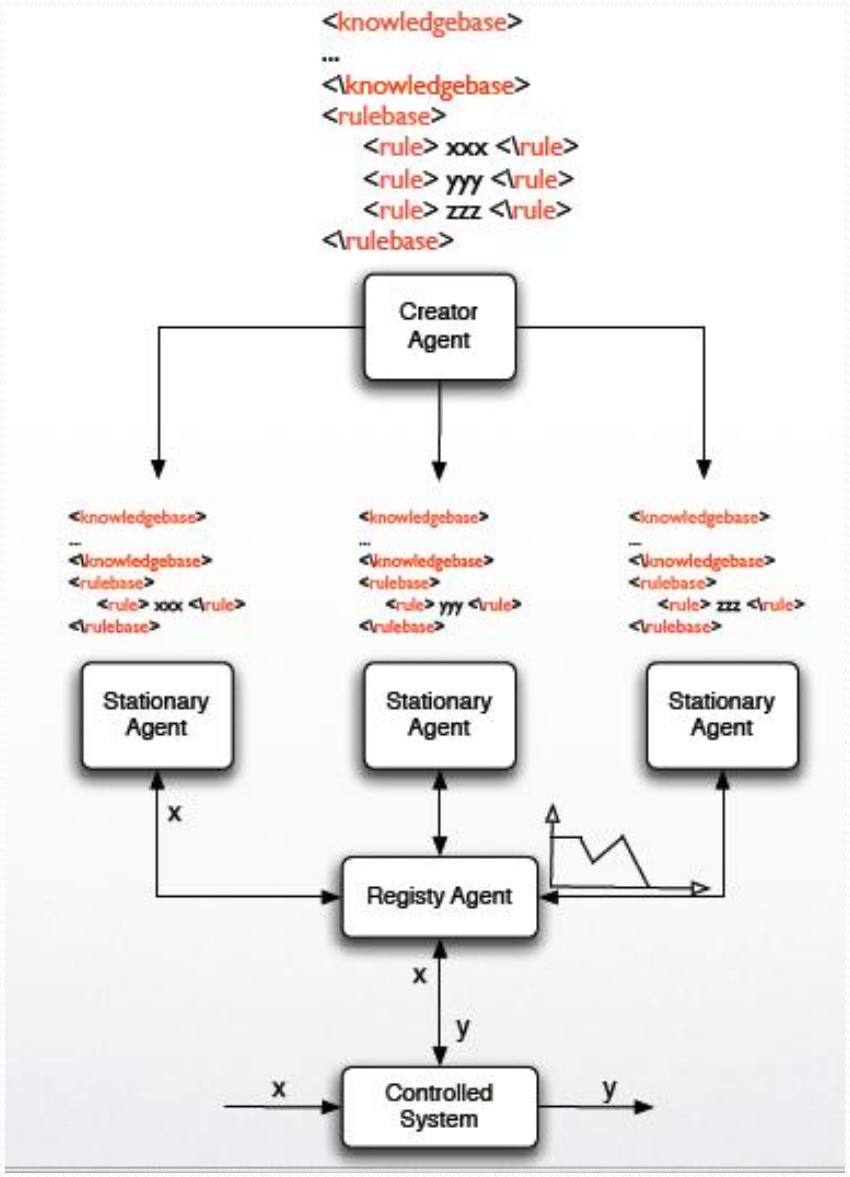
- It is possible to split FML tree structure into subtrees and place each one on a specific host.

- Advantages:
  - parallelize the fuzzy inference engine;
  - manage distributed knowledge environment;
  - exploit mobile agents as a natural and efficient technology to share data distribution and dispatch running code on a network.

Distributing FML

- **Application example**: an agent-based framework designed for providing proactive services in domotic environments.
- Ubiquitous devices can be used to parallel the fuzzy inference engine task by distributing fuzzy rules on them.
- FML program is a good model for rules distribution
  - It is simple to break FML code into many FML programs, each one containing a subset of FML rules.

Distributing FML program by means of Mobile Agents



Creator Agent is a software entities capable of reading FML code and breaking it into m FML programs, where m is the number of stationary agents living in the system.

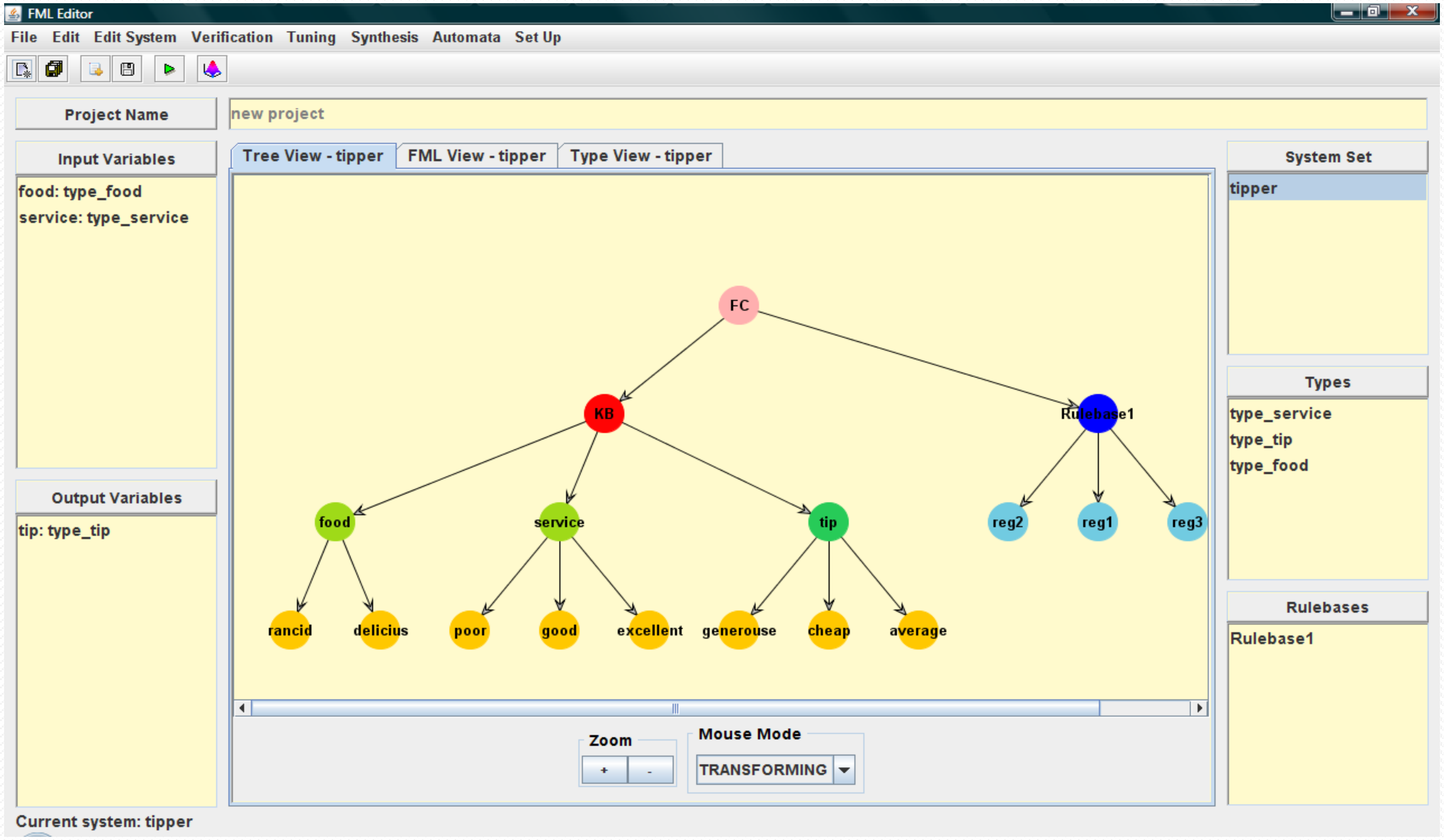
Stationary Agent are agents able to compute FML program by means of aforementioned technology (XSLT, JAXB, etc.)

Transport Agents are mobile entities moving from Registry Agent to Stationary Agents and vice versa. They transport the input/output values of fuzzy controller.

Registry Agent is the interface between controlled system and multi-agent system. It knows the Stationary Agents location and it uses Transport Agents to send input system values to Stationary Agents. Moreover, it compute a defuzzification method with values returned by Transport Agents.

An IDE for designing Transparent Fuzzy Agents

- The tree representation of a FLC and its mapping in FML language offers an additional important benefit: it allows to design and implement a fuzzy controller by means of simple visual steps.



Project Name: new project

Input Variables
food: type_food
service: type_service

Output Variables
tip: type_tip

System Set
tipper

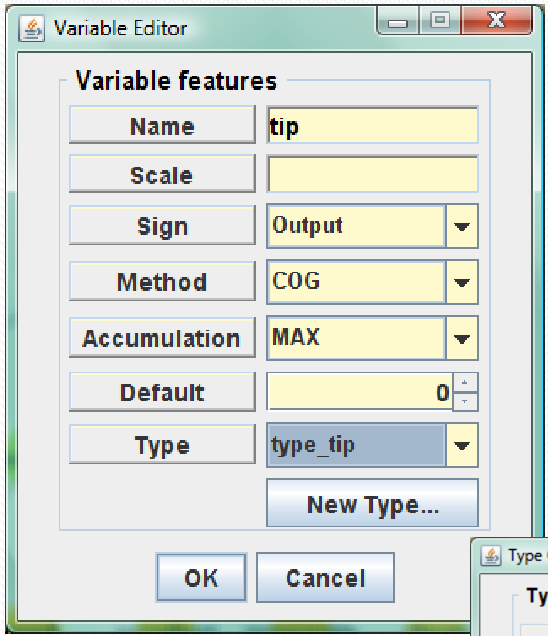
Types
type_service
type_tip
type_food

Rulebases
Rulebase1

Mouse Mode: TRANSFORMING

Current system: tipper

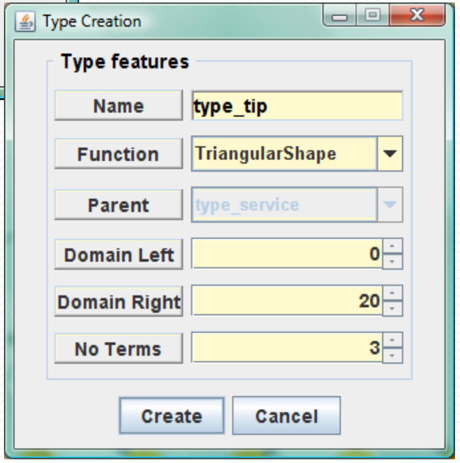
FML IDE – Creating a fuzzy variable



Variable Editor

Variable features

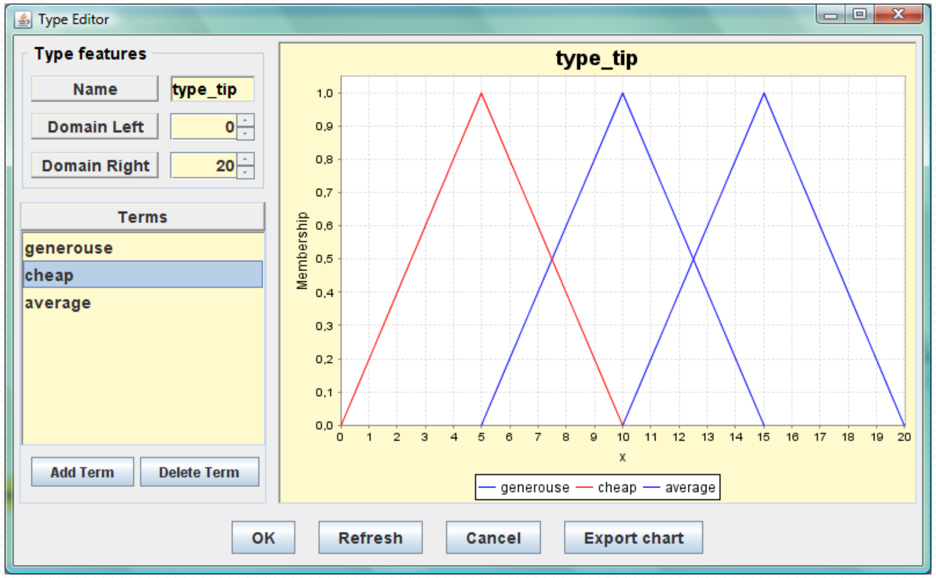
Name	tip
Scale	
Sign	Output
Method	COG
Accumulation	MAX
Default	0
Type	type_tip



Type Creation

Type features

Name	type_tip
Function	TriangularShape
Parent	type_service
Domain Left	0
Domain Right	20
No Terms	3



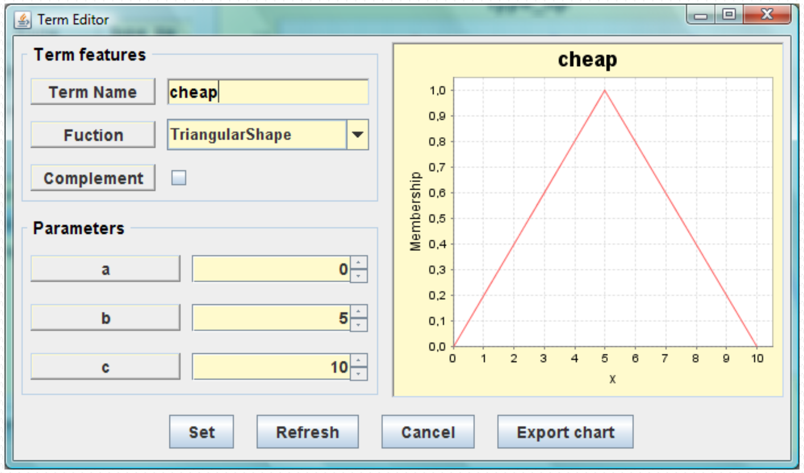
Type Editor

Type features

Name	type_tip
Domain Left	0
Domain Right	20

Terms

generouse
cheap
average



Term Editor

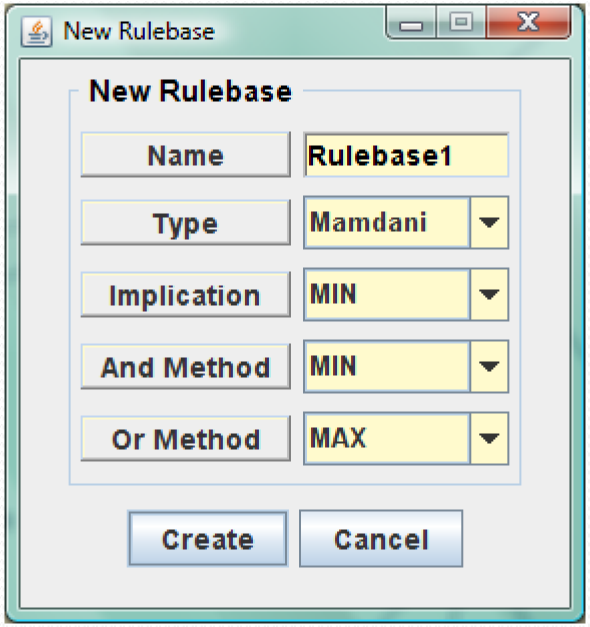
Term features

Term Name	cheap
Fuction	TriangularShape
Complement	☐

Parameters

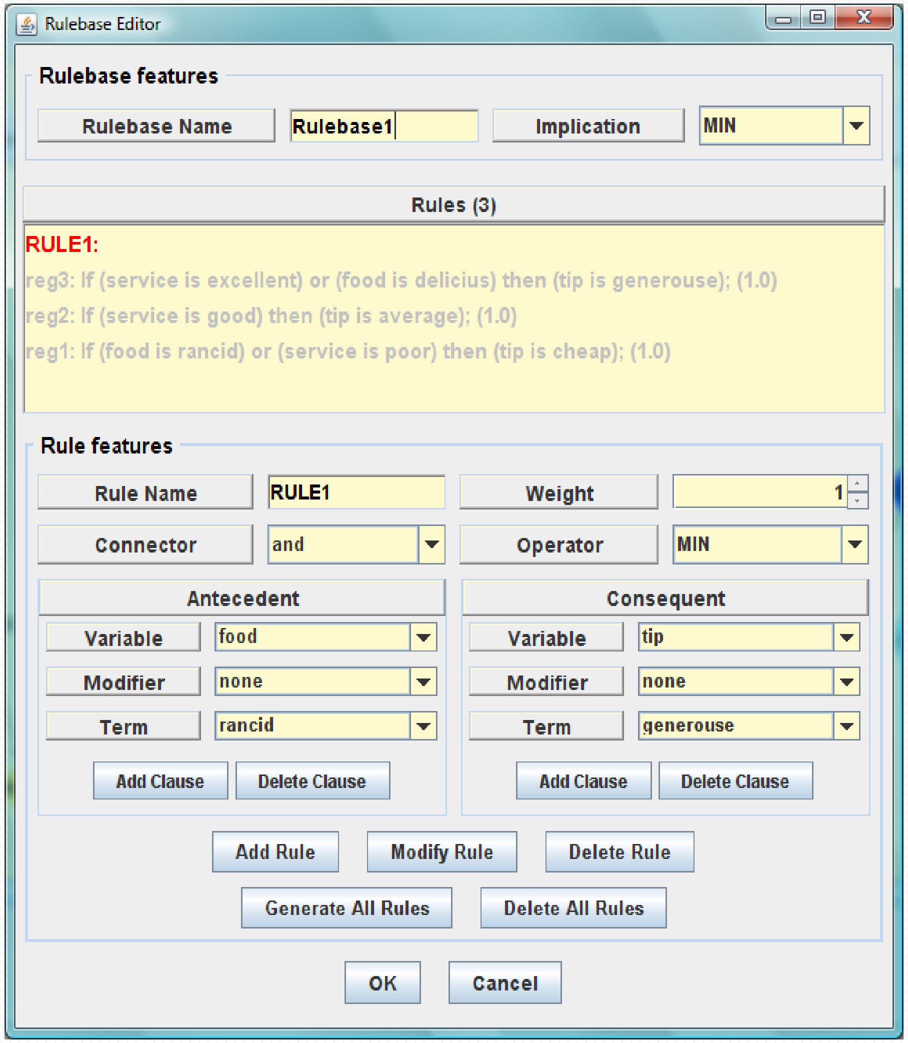
a	0
b	5
c	10

FML IDE– Creating a rule base


New Rulebase

New Rulebase

Name	Rulebase1
Type	Mamdani
Implication	MIN
And Method	MIN
Or Method	MAX


Rulebase Editor

Rulebase features

Rulebase Name	Rulebase1	Implication	MIN

Rules (3)

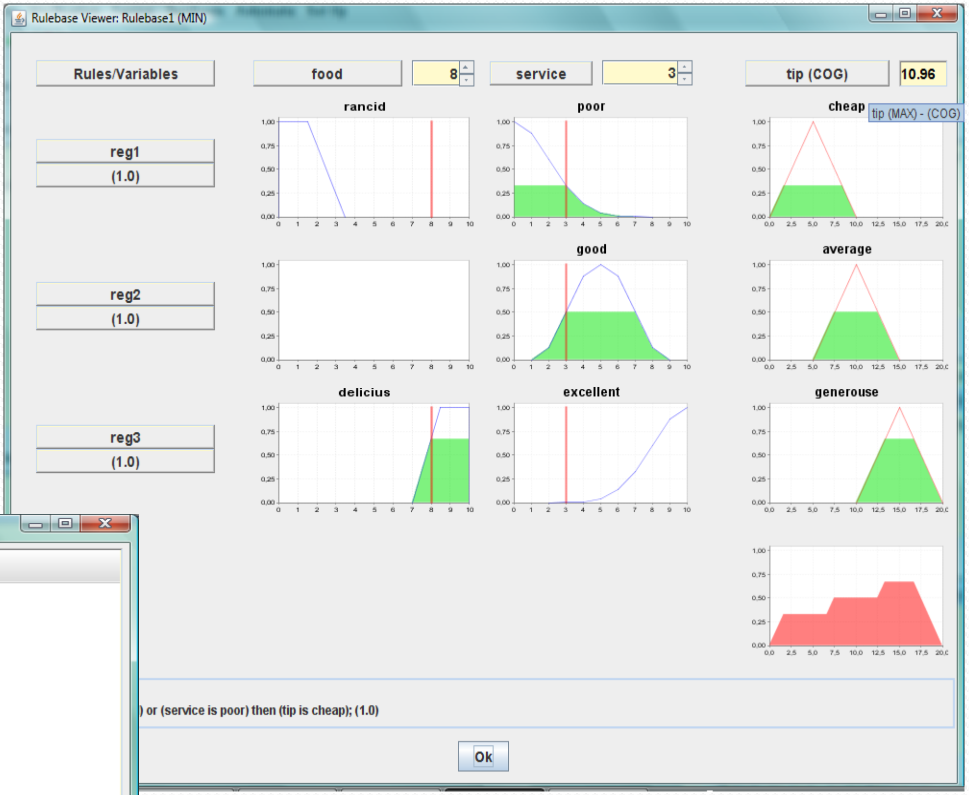
RULE1:
reg3: If (service is excellent) or (food is delicius) then (tip is generouse); (1.0)
reg2: If (service is good) then (tip is average); (1.0)
reg1: If (food is rancid) or (service is poor) then (tip is cheap); (1.0)

Rule features

Rule Name	RULE1	Weight	1
Connector	and	Operator	MIN

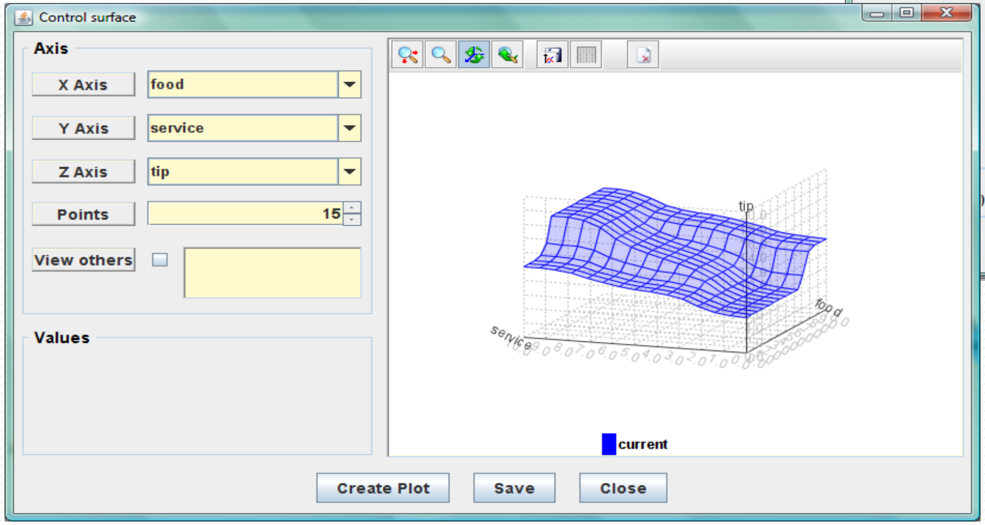
Antecedent		Consequent	
Variable	food	Variable	tip
Modifier	none	Modifier	none
Term	rancid	Term	generouse

FML IDE- Inference and Control Surfaces


Rulebase Viewer: Rulebase1 (MIN)

Rules/Variables	food	8	service	3	tip (COG)	10.96
reg1 (1.0)	rancid		poor		cheap	
reg2 (1.0)			good		average	
reg3 (1.0)	delicius		excellent		generouse	

) or (service is poor) then (tip is cheap); (1.0)


Control surface

Axis

X Axis	food
Y Axis	service
Z Axis	tip
Points	15
View others	☐

Values

FML Applications

- Ambient Intelligence
- Network Control
- Meeting Scheduling
- Computer Go
- Capability Maturity Model Integration (CMMI)
- Medical applications
  - Diet
    - Ontology-based Multi-AgentS (OMAS)
    - Type-2 Fuzzy Diet Assessment Agent
    - Intelligent Healthy Diet Planning Multi-agent
    - Ontology-based Intelligent Fuzzy Agent

FML Activities

- Special Session
  - IEEE WCCI 2010
  - Fuzz-IEEE 2011
  - IEEE WCCI 2012
- Special Issue
  - Springer Soft Computing Journal
- Other Editorial Activities
  - On the Power of Fuzzy Markup Language –
    Studies in Computational Intelligence - Springer

Conclusions

- A novel approach for fuzzy controllers design has been introduced.
- Based on XML
- Hardware Heterogeneity
- Multi-Agent Approach
- FML can be applied in different application scenarios

Thanks for Your Attention

IEEE WCCI 2016
Vancouver Canada

IEEE WORLD CONGRESS ON
COMPUTATIONAL INTELLIGENCE
24-29 JULY 2016, VANCOUVER, CANADA

FUZZ-IEEE 2016 Tutorials: FUZZ-IEEE-03
Part 3

Real-World Application on Game of Go

Yuandong Tian
Facebook AI Research, USA

K.W.S

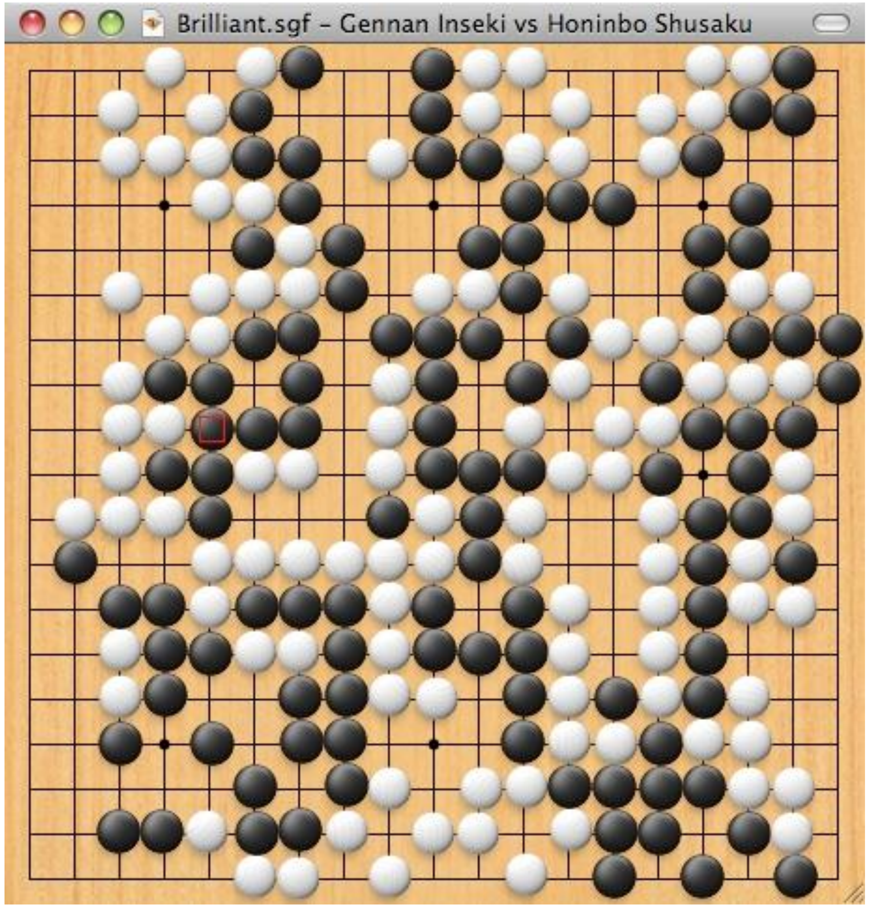
NUTN

DarkForest: An Open Source Computer Go engine

Yuandong Tian

Facebook AI Research

The Game of Go



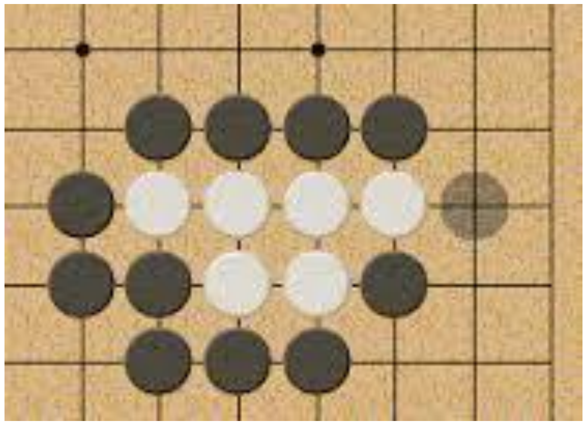
“A minute to learn, a lifetime to master”

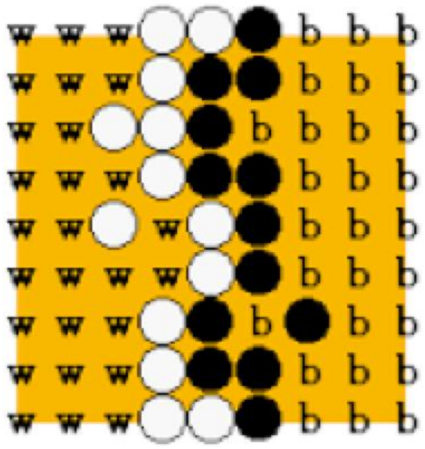
Rules

Black and white take turns on a 19*19 board.

(4-)connected group dies if surrounded by enemy.

The player with more territory wins.



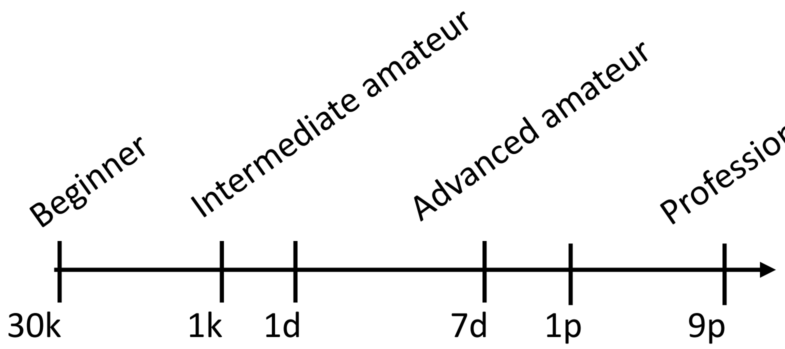


Why Go is interesting?

1. Unites Pattern Matching with Search
2. Combine Reason/Logic and Intuitions

Computer Go

- 50 years of Computer Go
  - Rule-based with alpha-beta pruning (1968-2005)
    - Kyu level
  - Monte-Carlo Tree Search (2006-2015)
    - 6d
  - Deep Convolutional Neural Network (2014-)
    - 6d -> Beyond 9p

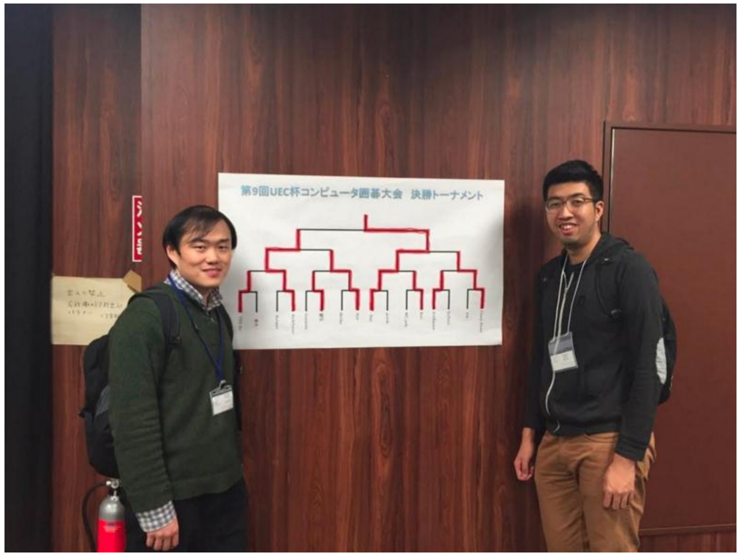


Overview of DarkForest

- Proposed by Yuandong Tian
- Developed by 2 people. Yuandong Tian and Yan Zhu
- Name after *The three body problem, Volume II, The Dark Forest*
- 1/100-1/1000 resource compared to AlphaGo

Strength of DarkForest

- Pure DCNN: KGS 3d, DCNN+MCTS: KGS 5d (stronger now)
- 3rd place on KGS January Tournaments
- 2nd place in UEC Computer Go Competition
- 4-GPU version: 6d-7d (tested by Chang-Shing Lee’s team)



History of DarkForest

- Data collection (May 2015)
- Pure DCNN on KGS (Aug 2015)
- MCTS working (Nov 2015)
- Distributed version (Dec 2015, Thanks Tudor Bosman!)
- Pachi’s default policy (Dec 2015)
- ICLR Accepted (Feb 2016)
- Learning-based default policy (Feb 2016, Thanks Ling Wang for Tygem dataset)
- Value network (July 2016)

Open Source

https://github.com/facebookresearch/darkforestGo

- License: BSD + PATENTS
- Self-made multithreaded Monte Carlo Tree Search
- Pretrained DCNN models (KGS 3d)
- Learning-based default policy
- Value network
- Training code to be released soon (using Torchnet)

How Go AI engine works

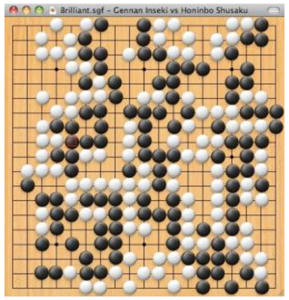
Even with a super-super computer,
it is not possible to search the entire space.

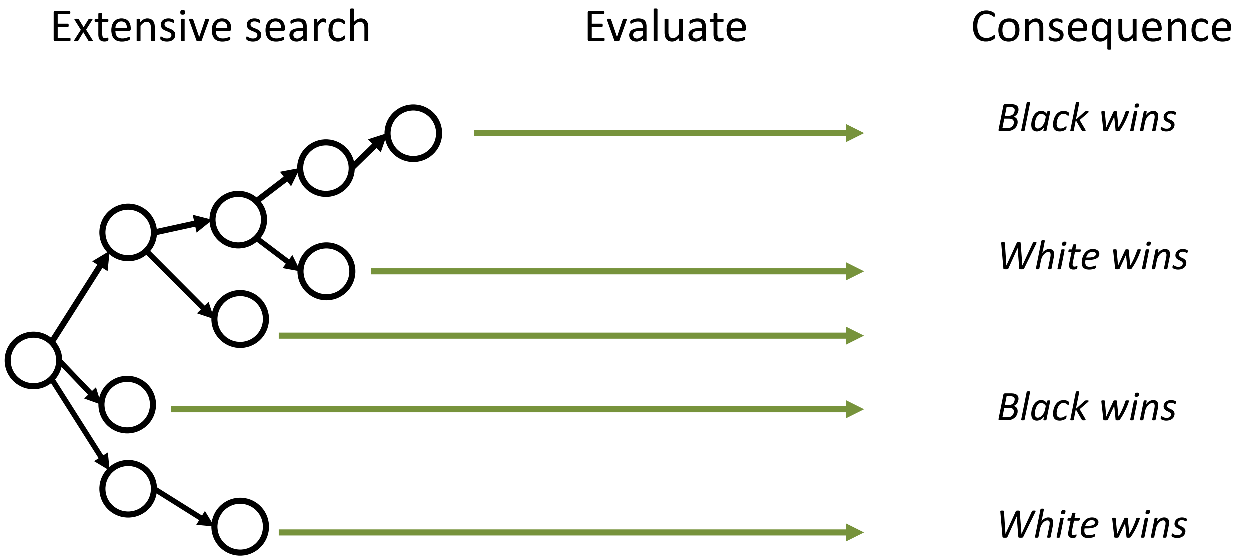
How Go AI engine works

Even with a super-super computer,
it is not possible to search the entire space.

How Go AI engine works

Even with a super-super computer,
it is not possible to search the entire space.

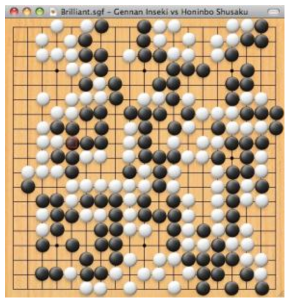

Current game situation



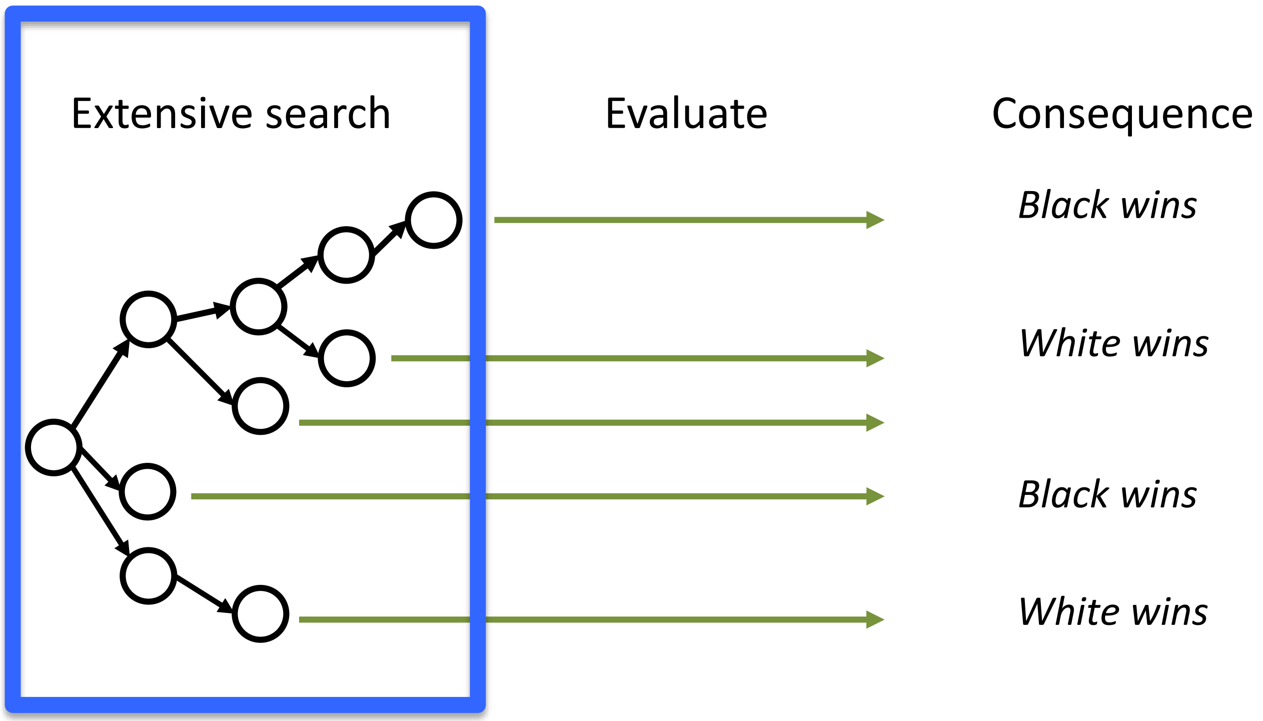
How Go AI engine works

How to expand a node? – Tree policy

Which node to expand? – Monte Carlo Tree Search

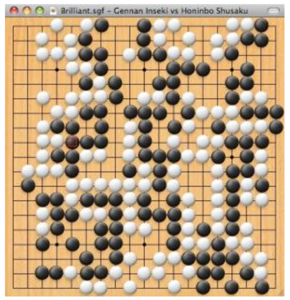


Current game situation

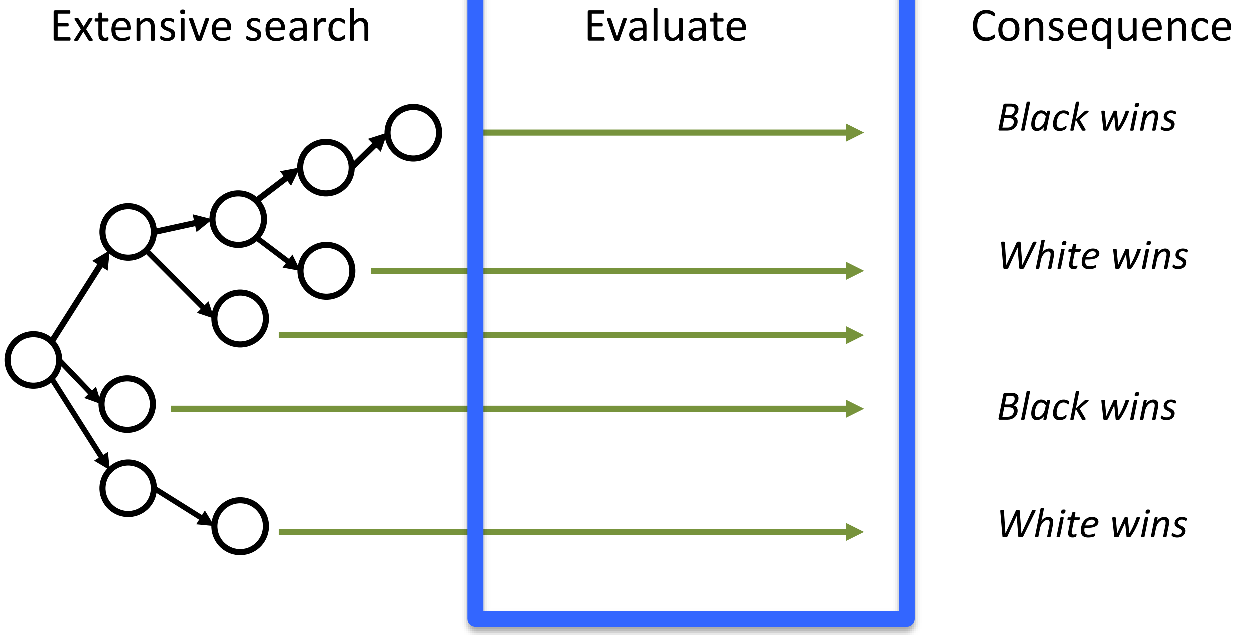


How Go AI engine works

How to evaluate? – Default Policy / Value function

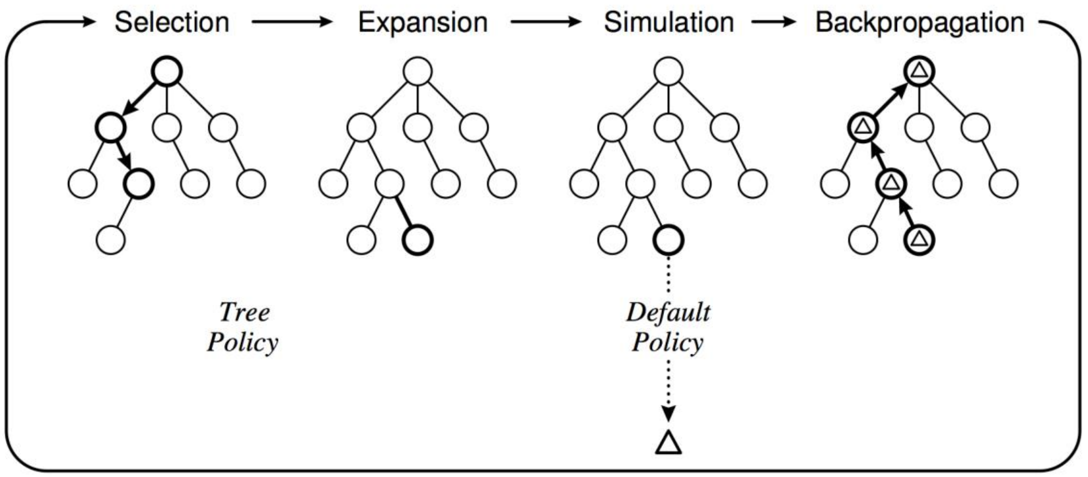


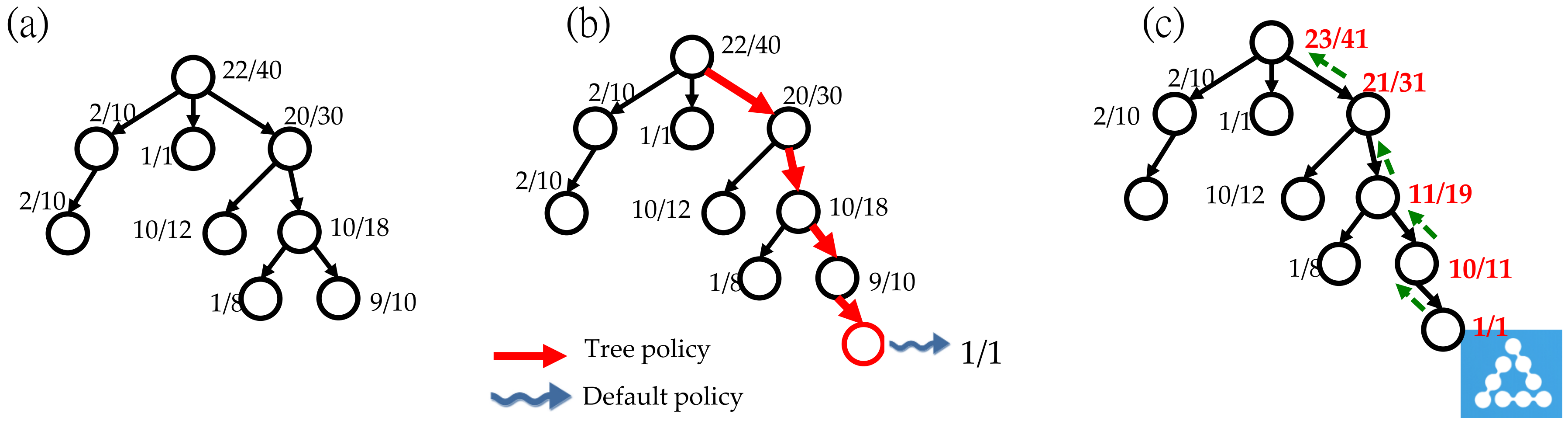
Current game situation



Monte Carlo Tree Search

- Aggregate win rates, and search towards the good nodes.





Why Go is Hard...

- Policy/Value function is hard to model
  - Chess: Summing over pieces, Go: ?
  - One stone difference completely changes the game.
- Traditional Heuristic Approach
  - Slow, hard to tune.
    - Pachi (open source go player) has lots of parameters to tune manually.
  - Conflicting parameters, not scalable.
  - Need strong Go experience.

Deep Learning can help!

- End-to-End training
  - No parameter tuning.
- Much less human intervention.
  - Minimal Go knowledge required.
- Amazing performance
  - Get the gist of the situation

Neural Network Attempts

- 1990s (small size and one hidden layer)
  - Not successful.
- University of Edinburgh, ICML 2015:
  - 4k-5k level
- Deepmind [ICLR 2015]
  - 12 layer CNN,beat Pachi (strongest open source AI) with 11% win rate.
- This paper [ICLR 2016]
- Deepmind Alpha Go [Nature 2016]

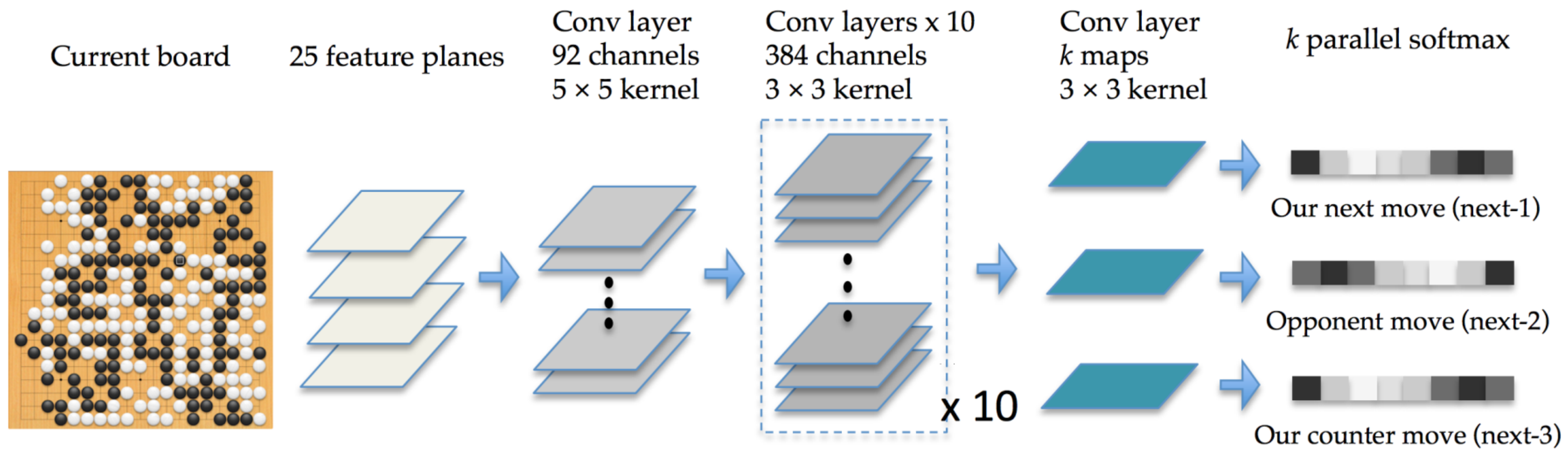
Overview of Architecture

Overview of Architecture

- DCNN training/testing
- Monte Carlo Tree Search (MCTS)
- Learning-based default policy

DCNN in DarkForest

- DCNN as a tree policy
  - Predict next k moves (rather than next move)
  - Trained on 170k KGS dataset/80k GoGoD, ***57.1%*** accuracy.
  - KGS 3D without search (0.1s per move)

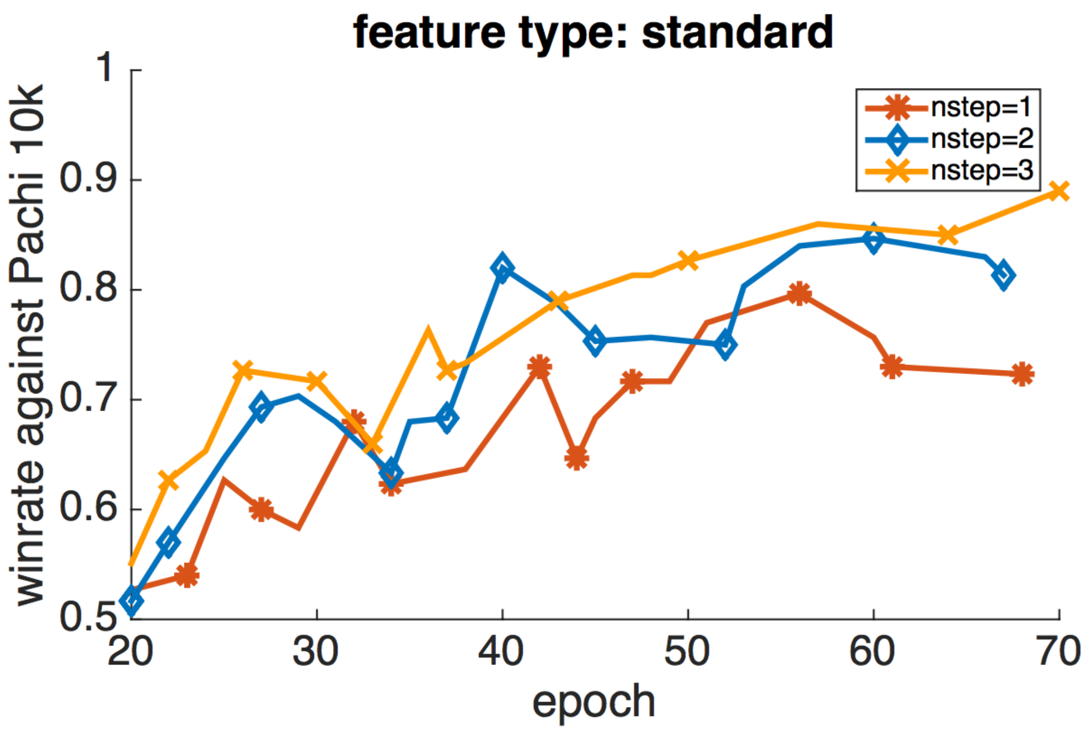


DCNN in DarkForest

- DCNN as a tree policy

Name
Our/enemy liberties
Ko location
Our/enemy stones/empty place
Our/enemy stone history
Opponent rank

Feature used for DCNN



Pure DCNN

darkforest: Only use top-1 prediction, trained on KGS
darkfores1: Use top-3 prediction, trained on GoGoD
darkfores2: *darkfores1* with fine-tuning.

	GnuGo (level 10)	Pachi 10k	Pachi 100k	Fuego 10k	Fuego 100k
Clark & Storkey (2015)	91.0	-	-	14.0	
Maddison et al. (2015)	97.2	47.4	11.0	23.3	12.5
darkforest	98.0 ± 1.0	71.5 ± 2.1	27.3 ± 3.0	84.5 ± 1.5	56.7 ± 2.5
darkfores1	99.7 ± 0.3	88.7 ± 2.1	59.0 ± 3.3	93.2 ± 1.5	78.0 ± 1.7
darkfores2	$\mathbf{100 \pm 0.0}$	$\mathbf{94.3 \pm 1.7}$	$\mathbf{72.6 \pm 1.9}$	$\mathbf{98.5 \pm 0.1}$	$\mathbf{89.7 \pm 2.1}$

Win rate between DCNN and open source engines.

Monte Carlo Tree Search

- Multi-threaded (use folly library)
- Block allocation for tree nodes
- Synchronized evaluation
  - Separate program for DCNN and MCTS
  - Communication: Linux Pipe (single machine) / Thrift (multiple machine)

Monte Carlo Tree Search

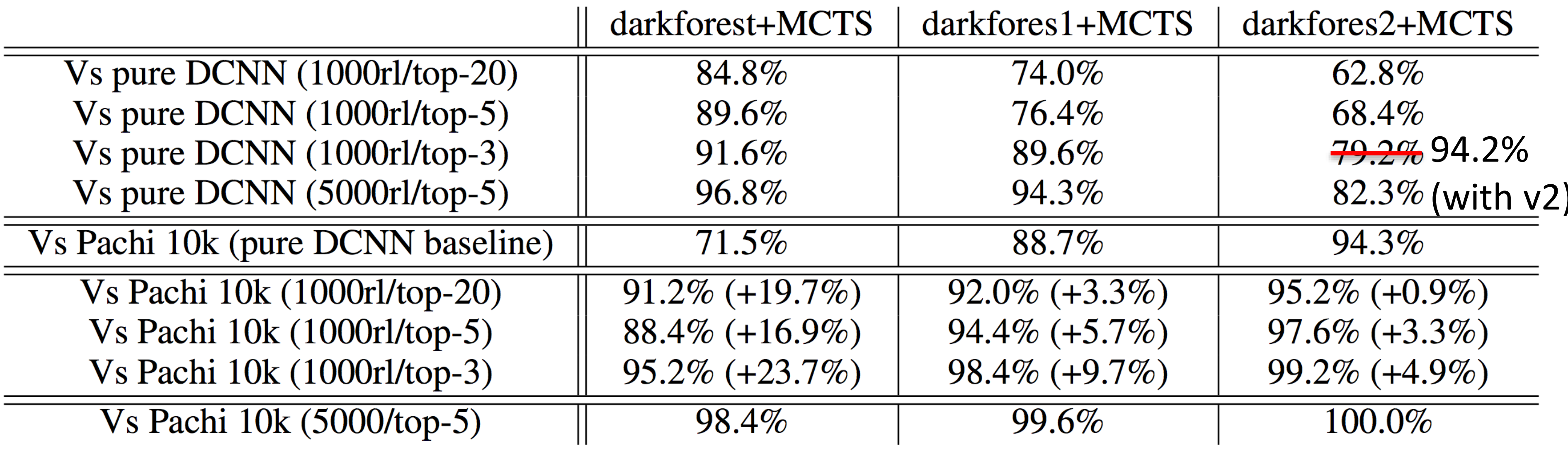
- December version
  - 100% synchronized.
  - Top3/5 moves from DCNN
  - Add noise to win rate
  - Use pachi default policy
- March version
  - 95% threads waiting until DCNN return/ 5% threads back to the root immediately
  - Virtual Counts: random 5 games at each new leaf
  - Use learning-based default policy
  - 85% win rate over previous version

Board evaluation

- Dead Stone Evaluation
  - Play default policy 100 times, stones are dead if they have low probability of survival
- Default policy
  - Rules: Save ours, attack opponents, play patterns, play nakade points, etc.
  - Learning:
    - Local 3x3 patterns hashed by Zobrist hashing
    - Keep a heap storing promising local 3x3 patterns.
  - Code includes: simple (only rules), pachi, v2 (full)

DCNN + MCTS

darkfmcts3: Top-3/5, 75k rollouts, ~12sec/move, KGS 5d

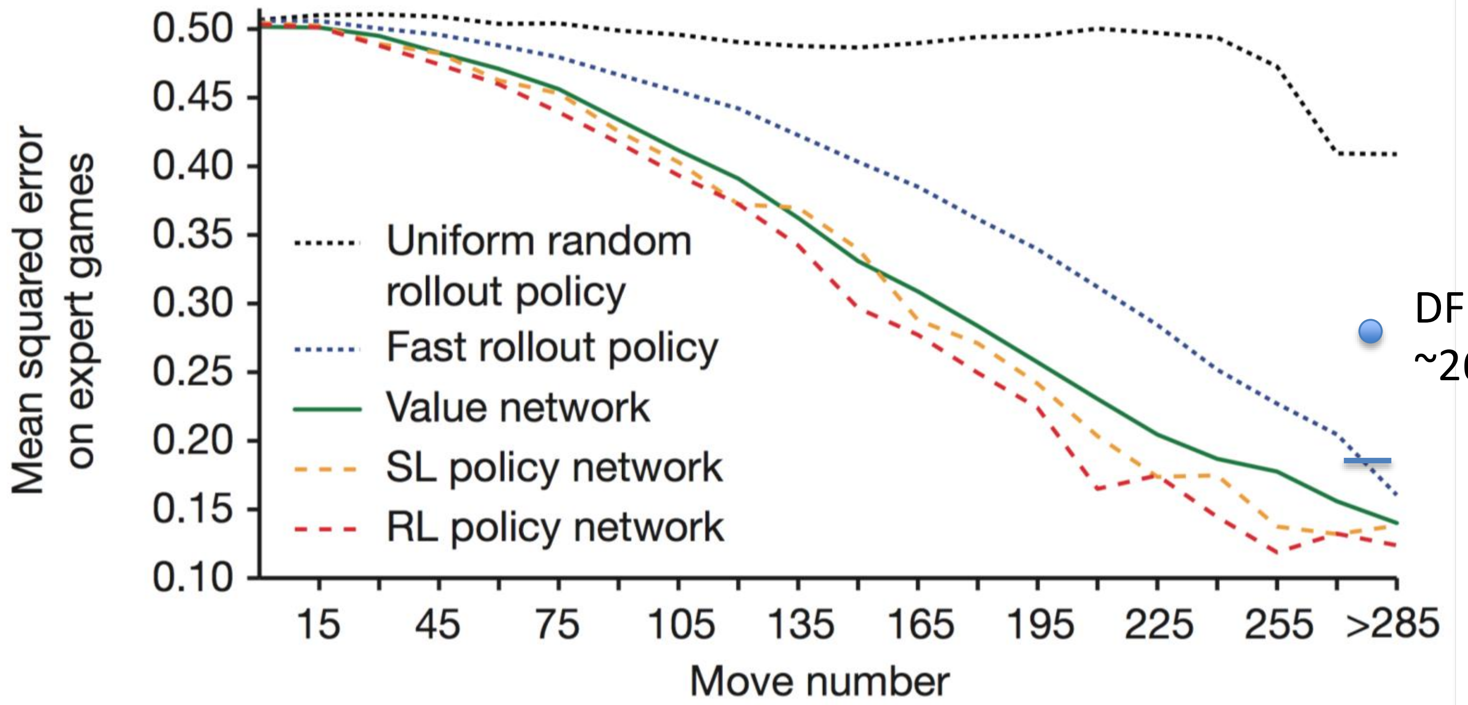


	darkforest+MCTS	darkfores1+MCTS	darkfores2+MCTS
Vs pure DCNN (1000rl/top-20)	84.8%	74.0%	62.8%
Vs pure DCNN (1000rl/top-5)	89.6%	76.4%	68.4%
Vs pure DCNN (1000rl/top-3)	91.6%	89.6%	~~79.2%~~ 94.2%
Vs pure DCNN (5000rl/top-5)	96.8%	94.3%	82.3% (with v2)
Vs Pachi 10k (pure DCNN baseline)	71.5%	88.7%	94.3%
Vs Pachi 10k (1000rl/top-20)	91.2% (+19.7%)	92.0% (+3.3%)	95.2% (+0.9%)
Vs Pachi 10k (1000rl/top-5)	88.4% (+16.9%)	94.4% (+5.7%)	97.6% (+3.3%)
Vs Pachi 10k (1000rl/top-3)	95.2% (+23.7%)	98.4% (+9.7%)	99.2% (+4.9%)
Vs Pachi 10k (5000/top-5)	98.4%	99.6%	100.0%

Win rate between DCNN + MCTS and open source engines.

Learning based default policy

- DF: 6 microsecond per move, ~30% accuracy.
  - However, Top-1 is not a good metric. Likelihood is
- Fig 2(b) in DeepMind's nature paper:



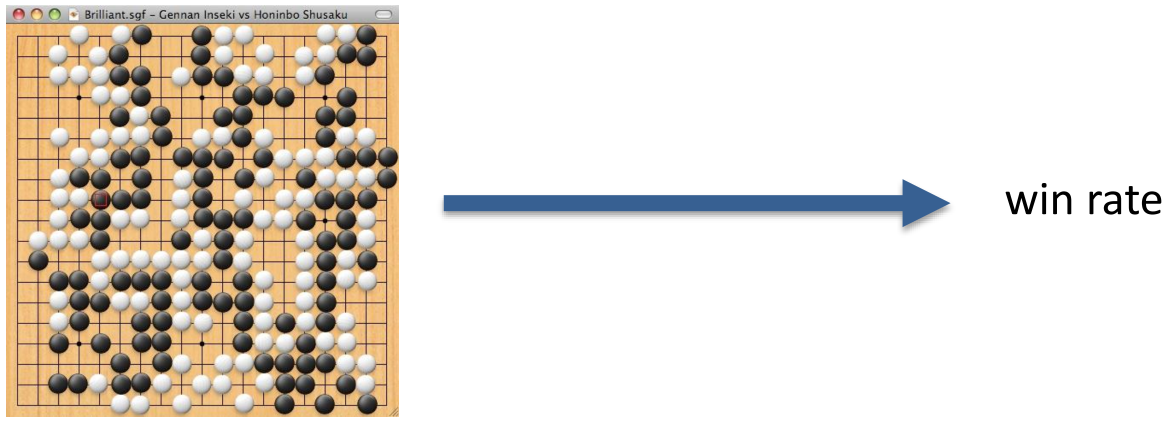
DF:
~26% in Move 280

Learning based default policy

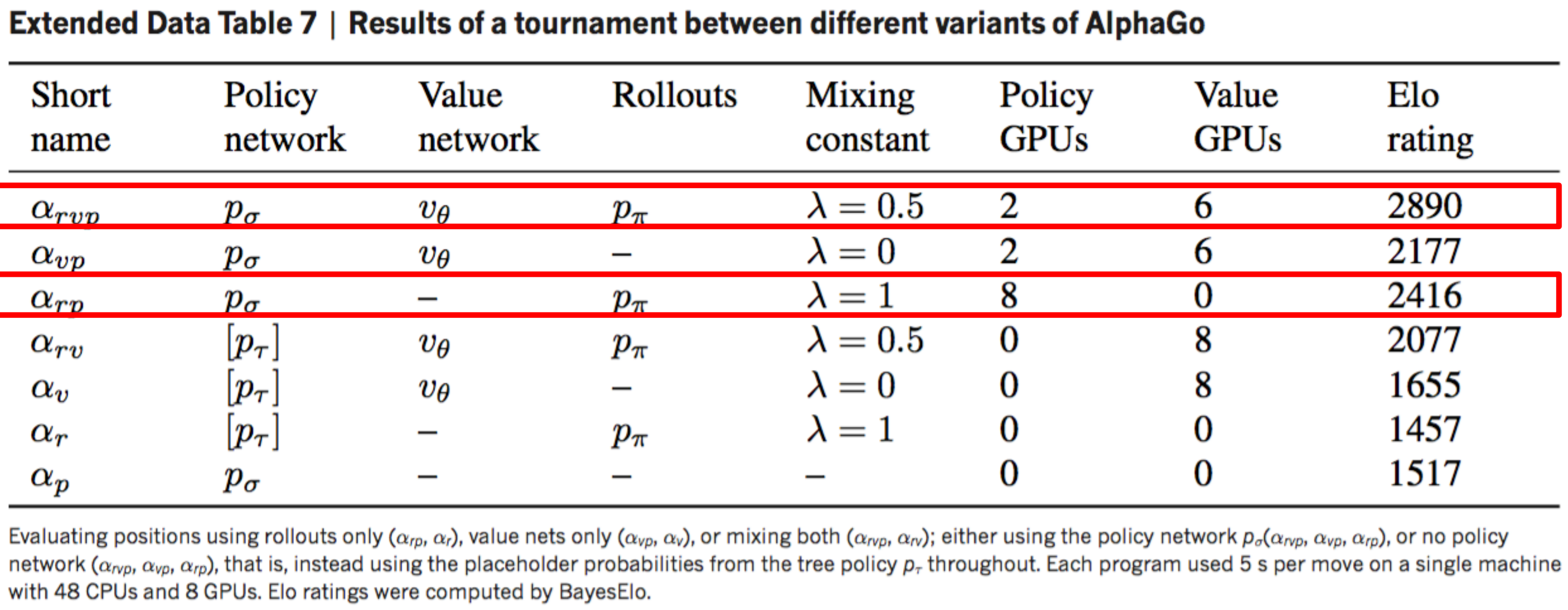
- How to close the gap?
  - Zen & CrazyStone spent 5-10 years in rules.
- Critical moves have to be 100% correct.
  - 2-point semeai.
  - Complicated Semeai on more than two groups.
  - Complicated life and death situations (corner & center)
- For now, no good way to learn automatically.
- How does AlphaGo solve this?

Value Network

- Board Evaluation
  - Use default policy:
    - Noisy/time consuming.
  - Use value function: fast!
    - Accuracy is the key
    - Hard to train



Value Network



Extended Data Table 7 | Results of a tournament between different variants of AlphaGo

Short name	Policy network	Value network	Rollouts	Mixing constant	Policy GPUs	Value GPUs	Elo rating
α_{rvp}	p_σ	v_θ	p_π	$\lambda = 0.5$	2	6	2890
α_{vp}	p_σ	v_θ	–	$\lambda = 0$	2	6	2177
α_{rp}	p_σ	–	p_π	$\lambda = 1$	8	0	2416
α_{rv}	$[p_\tau]$	v_θ	p_π	$\lambda = 0.5$	0	8	2077
α_{v}	$[p_\tau]$	v_θ	–	$\lambda = 0$	0	8	1655
α_{r}	$[p_\tau]$	–	p_π	$\lambda = 1$	0	0	1457
α_{p}	p_σ	–	–	–	0	0	1517

Evaluating positions using rollouts only (α_{rp}, α_r), value nets only (α_{vp}, α_v), or mixing both (α_{rvp}, α_{rv}); either using the policy network p_σ(α_{rvp}, α_{vp}, α_{rp}), or no policy network (α_{rvp}, α_{vp}, α_{rp}), that is, instead using the placeholder probabilities from the tree policy p_τ throughout. Each program used 5 s per move on a single machine with 48 CPUs and 8 GPUs. Elo ratings were computed by BayesElo.

+500 Elo (~2 stones) for AlphaGo

My guess:
+ Playout/Simulation is good at local battle in complicated situations.
+ Value network is good at global reading, saving thousands of simulations

Value Network

- We made some progress (~0.15 MSE)
  - Generate 1.2M self-play games with DF+DF2
    - Similar approach with AlphaGo
    - DF for more diverse moves
    - DF2 for precise moves (for better end-game evaluation)
  - Initialize the weights of last few layers with DF2.
  - Adagrads works very well.

Thanks!