الجامعة الافتراضية السورية

SYRIAN VIRTUAL UNIVERSITY

النظم الخبيرة

الدكتورة أميمة دگاك

Books

النظم الخبيرة

الدكتورة أميمة دكّاك

من منشورات الجامعة الافتراضية السورية

الجمهورية العربية السورية 2018

Expert Systems

Oumaima Dkkak

Publications of the Syrian Virtual University (SVU)

Syrian Arab Republic, 2018

الفهرس

١)مدخل الى النظم الخبيرة..... ١
١. النظم الذكية..... ٢
٢. نظم قواعد المعرفة..... ٣
٣. قاعدة المعرفة وطرق المحاكمة..... ٤
٤. النظم الخبيرة..... ٩
٥. تحصيل المعرفة..... ١٠
٦. الذكاء الحوسبي..... ١١
٢)خصائص النظم الخبيرة..... ١٢
١. النظم الخبيرة..... ١٣
٢. النظم المعتمدة على قواعد المعرفة..... ١٤
٣. النظم المقودة بالمعطيات..... ١٦
٤. ميزات خاصة بالنظم العائمة..... ١٧
٥. نظم خبيرة للتحكم والمحاكمة..... ١٨
٦. خواص النظم الخبيرة الناجحة..... ٢٠
٣)النظم الخبيرة المعتمدة على القواعد..... ٢٣
١. الحقائق والقواعد..... ٢٤
٢. السلسلة الأمامية والسلسلة الخلفية..... ٢٦
٣. حل التعارض..... ٣٠
٤. معالجة الشك..... ٣٠
٤)النظم الخبيرة المعتمدة على الأطر..... ٣٤
١. الأغراض والأطر والبرمجة غرضية التوجه..... ٣٥
٢. تجريد المعطيات..... ٣٥
٣. الوراثة ٤١
٤. الكبسلة..... ٤٦
٥. لغة النمذجة الموحدة UMLوالربط الديناميكي..... ٤٧
٦. مواضيع أخرى..... ٤٩
٥)المنطق العائم ٥٣
١. المنطق التقليدي والمنطق العائم..... ٥٤
٢. المجموعات العائمة..... ٥٧
٣. العلاقات العائمة..... ٦٠
٤. الفرضيات العائمة..... ٦١
٥. التعويم وفك التعويم..... ٦٢
٦. جبر المجموعات العائمة..... ٦٦

٧. الحساب العائم.....٦٨
٦) النظم الخبيرة العائمة.....٧٢
١. أنواع الاستدلال العائم.....٧٤
٢. طبيعة الاستدلال في النظم الخبيرة العائمة.....٧٤
٣. المحاكمة التقريبية.....٧٧
٧) دراسة حالات وأمثلة على النظم الخبيرة.....٨٢
١. نظم التخطيط.....٨٣
❖ نظم التخطيط التقليدية.....٨٣
❖ STRIPS.....٨٤
❖ التخطيط التراتبي(الهرمي) ٨٧
❖ تأجيل الضمانات.....٨٨
❖ الجدولة.....٨٩
❖ التحليل المعتمدعلى القيود.....٨٩
٢. نظم التحكم.....٩١
❖ التحكم المنخفض المستوى.....٩١
❖ متطلبات التحكم عالية المستوى.....٩٢
❖ المحاكمة المقيدة زمنيا.....٩٣
❖ التحكم العائم.....٩٣
❖ متحكم BOXES.....٩٤
❖ التحكم الإحصائي بالإجرائيات.....٩٥
٨) المراجع.....٩٦

الفصل الأول

مدخل إلى النظم الخبيرة

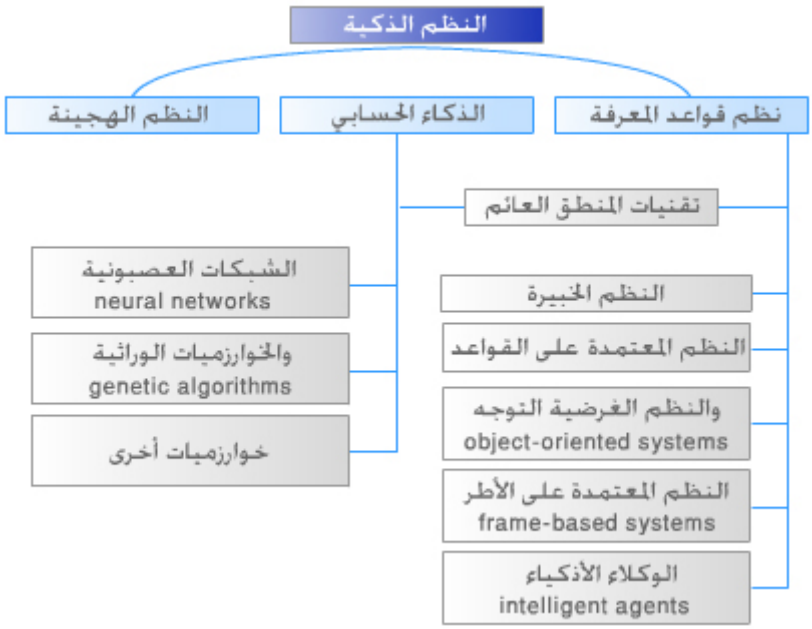
1. النظم الذكية

منذ عدة قرون، طور الإنسان مجموعة أدوات لخدمته. أدوات كثيرة كان لها دور هام في تاريخ حضارته: الرمح والمطرقة والبندقية والسيارة والطائرة وغيرها. كما طور أدوات تواصل مثل اللغة المنطوقة والمكتوبة والرياضيات. ساعدت هذه الأدوات على خزن المعلومات وتبادلها وعلى التعبير عن المفاهيم.

كما شهدت العقود الأخيرة ولادة الحاسوب الرقمي. باستطاعة الحاسوب إنجاز عمليات حسابية ورمزية يمكن للإنسان العادي القيام بها، ويمتاز الحاسوب بسرعة تنفيذ هذه العمليات ودقتها. ولكن هل استطاع الإنسان بناء حاسوب يمكنه التفكير؟

تهدف بحوث الذكاء الصنعي إلى بناء آلة كهذه تفكر، وإلى تحسين فهمنا للذكاء. والهدف الأسمى في هذا الحقل هو بناء آلة تستطيع أن تقلد الإمكانات العقلية لدى الإنسان أو تتفوق عليها، من حيث المحاكمة والفهم والتخيل والإدراك والإبداع والعواطف. لازلنا بعيدين عن هذا الهدف، ولكن لدينا آلات يمكنها تقليد الإنسان في لعب الشطرنج وتركيب الكلام وتشخيص الأمراض وتعرف الصور وفي مهام أخرى عديدة.

ثمة أدوات ذات أهمية خاصة في هذا المجال، يمكن تقسيمها إلى ثلاثة أصناف: نظم قواعد المعرفة والذكاء الحسابي والنظم الهجينة. تتضمن نظم قواعد المعرفة النظم الخبيرة والنظم المعتمدة على القواعد، والنظم الغرضية التوجه **object-oriented systems**، والنظم المعتمدة على الأطر **frame-based systems**، والوكلاء الأذكياء **intelligent agents**. أما الذكاء الحسابي **intelligent computation** فيضم الشبكات العصبونية **neural networks** والخوارزميات الوراثية **genetic algorithms** وخوارزميات أمثلة أخرى.

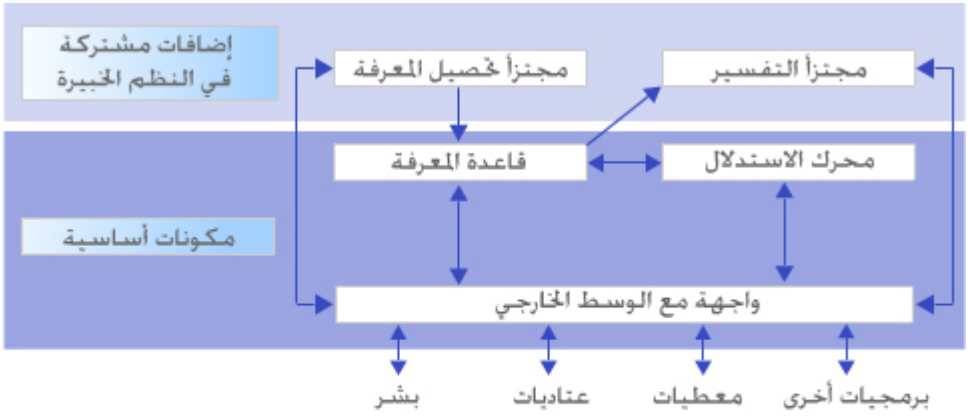


وتندرج تقنيات المنطق العائم تحت كلا الصنفين. ونشير إلى الأصناف الثلاثة مجتمعة بالنظم الذكية.

لم تحل النظم الذكية مشكلة بناء عقل صنعي، ويعتقد البعض أنها تمتلك قدراً ضئيلاً من الذكاء إن وجد. ولكن بالمقابل، فإن هذه النظم ساهمت في حل العديد من المسائل التي كانت تبدو صعبة ومعقدة من دونها. ومن هنا تأتي أهميتها التطبيقية.

2. نظم قواعد المعرفة

يكمن الفرق بين نظم قواعد المعرفة والبرامج العادية في البنيان **structure**. ففي البرنامج العادي، نجد المعرفة وأدوات التحكم فيها موجودة معاً جنباً إلى جنب، وأي تعديل في أي منها يتطلب تعديلاً في البرنامج. أما في نظم قواعد المعرفة فالدوران منفصلان تماماً. وفي أبسط الحالات لدينا مجتزآن **two modules**: مجتزأ المعرفة ويسمى قاعدة المعرفة **knowledge base**، ومجتزأ التحكم ويسمى محرك الاستدلال **inference engine** (انظر الشكل 1). وفي بعض النظم المعقدة، يمكن أن يكون محرك الاستدلال نفسه نظاماً معتمداً على القواعد، يتضمن معرفة مترفعة **meta-knowledge**؛ أي معرفة حول المعرفة، وهنا هي معرفة حول كيفية تطبيق معرفة النطاق **domain knowledge** الذي نحاول حل المسألة فيه.



الشكل -1- المكونات الأساسية لنظام معتمد على القواعد

يسهل، في هذه النظم، فصل المعرفة عن التحكم، أو إضافة معارف جديدة، إما أثناء تطوير البرنامج أو في ضوء الخبرة التي نكتسبها أثناء حياة البرنامج. في هذا الأمر محاكاةٌ لعمل العقل البشري، فإجرائيات التحكم (محرك الاستدلال) تبقى ثابتة بطبيعتها، مع أن سلوك الفرد يتغير باستمرار مع ازدياد الخبرة والمعرفة (تحديث قاعدة المعرفة).

لنفترض أن مهندساً متمرساً يستخدم برنامجاً عادياً يساعده في عمله اليومي. يتطلب تغيير سلوك البرنامج أن يكون المهندس على دراية بتفاصيل البرنامج. سيتطلب هذا نموذجياً تعديل عبارات تحكم من الشكل:

If…then…else…

أو من الشكل:

For x from a to b do…

ولإجراء هذه التعديلات، يجب أن يكون المهندس مبرمجاً متمرساً. حتى وإن كان كذلك، فإن مثل هذه التعديلات صعب لأنه قد يؤثر في سلوك البرنامج في أماكن أخرى منه، على نحو غير مرغوب.

إن منهج نظم قواعد المعرفة سهل ومباشر. المعرفة ممثلة صراحة في قاعدة المعرفة وليس ضمنياً في بنيان البرنامج. يمكن تغيير هذه المعرفة بسهولة في قاعدة المعرفة، ويستخدم محرك الاستدلال قاعدة المعرفة لمعالجة مهمة خاصة بالطريقة التي يستخدم فيها البرنامج العادي ملف معطيات.

3. قاعدة المعرفة وطرق المحاكمة

يمكن أن تكون قاعدة المعرفة غنية جداً بالمعارف. سنفترض هنا أنها تتضمن ببساطة مجموعة قواعد ومجموعة حقائق. يمكن أن تكون القواعد معقدة، ويمكن أن تتضمن الحقائق سلاسل وبنى مهيكلة **structures** وواصفات لهذه الكيانات، وعلاقات بينها. ويمكن أن تتغير تفاصيل تمثيل الحقائق من نظام إلى آخر. لنأخذ على سبيل المثال نظاماً يتضمن حقائق من النمط التالي:

"جو يعمل في شركة **ACME**"

Joe works for ACME

وقواعد من نمط:

"إذا كان **?x** يعمل في شركة **ACME** فلديه مرتب شهري كبير"

IF ?x works for ACME THEN ?x earns a large salary.

تشير إشارة الاستفهام ? إلى أن ما بعدها متغير، يمكن الاستعاضة عنه بأي قيمة ثابتة مثل "جو" أو "ماري".

لتمثيل الحقيقة والقاعدة السابقتين، في برنامج عادي، يمكننا أن نبدأ بتعريف **record** (بنيان معطيات يجمع عدة أنماط من المعطيات) لكل موظف. ونمثل القاعدة باستخدام عبارة **IF...THEN...** ولكن يجب وضع هذه العبارة في المكان المناسب من البرنامج، بعناية، بحيث:

- يجري استخدام العبارة حيثما احتجنا إليها.
- يمكن الوصول إلى جميع المتغيرات البارزة فيها.
- القيم المعطاة للمتغيرات تبقى متاحة طالما احتجنا إليها.

في الحقيقة، كأن الحقائق والقواعد مرتبطة عتادياً، وهي جزء لا يتجزأ من البرنامج. وبما أن معظم النظم تحتاج إلى مئات بل آلاف القواعد والحقائق، فإن إدراج هذه المعارف في البرامج أمر صعب. بالمقابل في نظم قواعد المعرفة، يمكن تعديل الحقائق والقواعد بسهولة حسب الرغبة، لأنها ممثلة تمثيلاً صريحاً.

بقي أن نشير إلى أننا إلى الآن، نفترض أننا نتعامل مع معارف أكيدة، وهذا ليس محققاً دائماً. سنناقش لاحقاً الشك في القواعد، وهو يمكن أن ينشأ من ثلاثة أمور هي:

- شك في الدليل **evidence**.[1]
- شك في علاقة الدليل بالنتيجة.[2]
- قاعدة غامضة.[3]

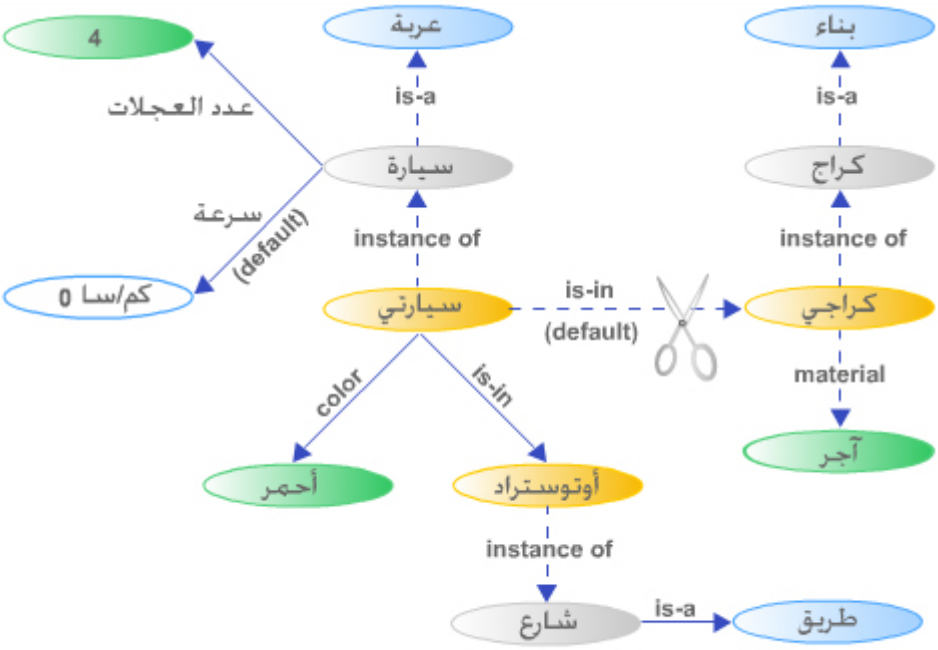
يمكن أن تكون الحقائق سكونية **static**، كالحقيقة في المثال السابق، وعندها يمكن أن تُكتب في قاعدة المعرفة. ولا يعني هذا ألا تتغير الحقائق، ولكن تغيراتها بطيئة بحيث يمكن تحديثها ضمن قاعدة المعرفة. بالمقابل، يمكن أن تكون الحقائق عابرة، مثل: "ضغط الغاز **3000** باسكال". هنا هذه الحقيقة نتيجة قياس لحظي قد يتغير. يمكن إعطاء قيمة مغتفلة **by default** تُستخدم في غياب القياسات.

[1] قد لا نكون متأكدين من أن جو يعمل في شركة ACME

[2] قد لا نكون متأكدين أن رواتب جميع موظفي الشركة عالية

[3] ما معنى أن يكون الراتب عالياً؟

لننظر في المثال التالي، الذي نمثل فيه المعارف على شكل شبكة دلالة



الشكل -2- شبكة دلالة مع قيم مغتفلة default

هنا بعض العلاقات سكونية: سيارتي هي سيارة، السيارة هي عربة، كراجي هو كراج، الكراج هو بناء، الأوتوستراد هو شارع، والشارع هو طريق.

وبعض الواصفات سكونية: عدد عجلات السيارة 4، لون سيارتي أحمر، والمادة المصنوع منها كراجي هي الآجر.

أما القيم المغتفلة فهي:

- قيمة حقيقة "سرعة سيارتي" وهي 0 كم/سا،
- والعلاقة: "سيارتي في كراجي".

وتبقى العلاقة: "سيارتي في الأوتوستراد"، علاقة عابرة.

نلاحظ أننا ميزنا بين كينونتين **entities**:

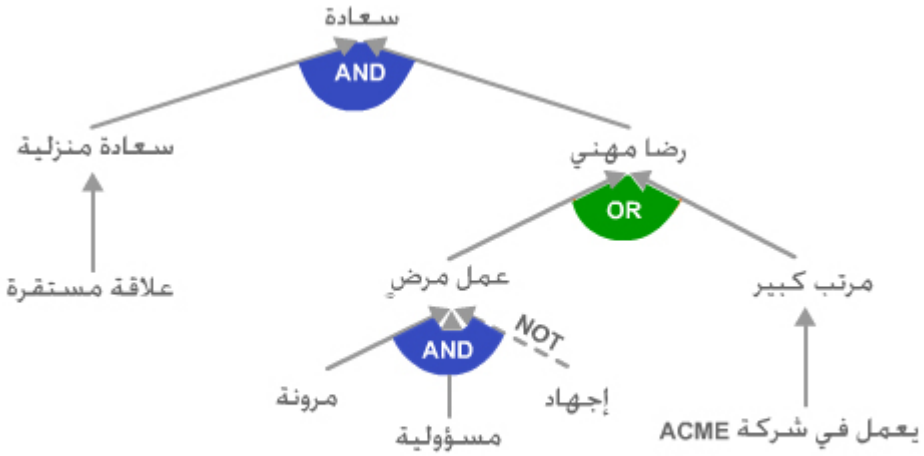
الواصفة **attribute** والعلاقة **relation**.

وقد مثلنا العلاقات بين المنتسخات **instances**

والواصفات للأغراض في شكل بياني يسمى "شبكات الدلالة" **semantic net**.

(انظر الشكل **-2-**)

الحقائق السكونية أو العابرة متاحة في قاعدة المعرفة. ويمكن استنتاج معارف أخرى بتطبيق القواعد القابلة للتطبيق، والتي يمكن أن تكون مستقلة (أي يمكن أن تكون نتيجة قاعدة ما، شرطٌ لقاعدة ثانية). يمكن أن تُعرف العلاقات بين القواعد شبكةً نسميها شبكة الاستدلال **inference net** كما في الشكل التالي:



الشكل -3- شبكة استدلال

الحقائق المشتقة نتيجة تطبيق قواعد يمكن أن تحقق شروط قاعدة أخرى، كلياً أو جزئياً. يشكل مجموع القواعد شبكة الاستدلال، وهي تُستخدم بدورها لتربط بين المسببات والنتائج **IF <cause> THEN <effect>.**

من الشبكة المبينة بالشكل السابق يمكن استنتاج مايلي: إذا كان "جو" يعمل في شركة **ACME** ويعيش علاقة مستقرة (السبب)، فهو سعيد (النتيجة). تسمى هذه العملية بالاستنتاج **deduction.**

بعض المسائل مثل تشخيص الأمراض، تتطلب محاكمة بالاتجاه المعاكس، إذ نتمنى أن نعرف السبب من النتائج، وتسمى هذه العملية الاستقراء **abduction**. فمن معرفة أن جو سعيد، وبالاستقراء نستنتج أن لديه رضى مهني وسعادة منزلية. ولكن، هذا الاستنتاج لايكون صالحاً إلا إذا أظهرت شبكة الاستدلال كل الطرق التي تؤدي إلى سعادة شخص ما. هذا الافتراض هو افتراض العالم المغلق، حيث لانعرف أي شيء خارج حدود الشبكة. في هذه الحالة، إذا طرأ تغير على أحد العقد (كأن يغير جو عمله) فقد يتأثر الكثير من العقد الأخرى.

من جهة أخرى، إذا كانت لدينا عدة أمثلة **instances or examples**، نعرف أسبابها ونتائجها، يمكننا استدلال قاعدة، أو شبكة استدلال. على سبيل المثال، إذا كانت رواتب كل من قابلناهم من شركة **ACME** كبيرة فيمكننا استنتاج القاعدة التالية:

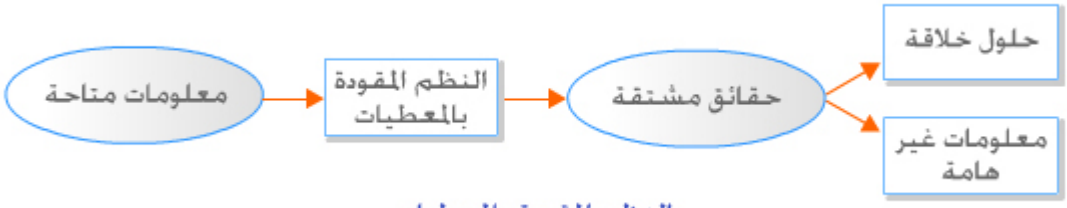
"إذا كان x يعمل لدى شركة ACME فإن مرتبه كبير"

IF ?x works for ACME THEN ?x earns a large salary.

تسمى عملية استدلال القاعدة **induction**.

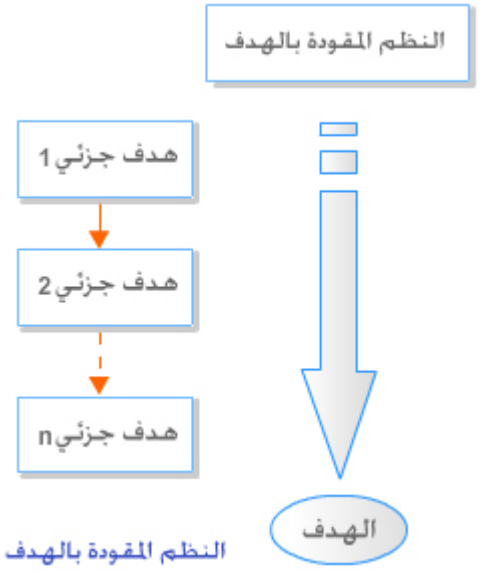
يتغير محرك الاستدلال **inference engine** بحسب نوع المعرفة التي نتعامل معها وتعقيدها. يمكننا تمييز نوعين من محركات الاستدلال: ذات السلسلة الأمامية **forward chaining** وذات السلسلة الخلفية **backward chaining**. وتسمى هذه المحركات أحياناً المحركات المقودة بالمعطيات **data driven** والمحركات المقودة بالأهداف **goal driven** على التتالي.

النظم المقودة بالمعطيات تأخذ المعلومات المتاحة وتولد منها حقائق مشتقة، قدر المستطاع. وهذا إما أن يقود إلى حلول خلاقة للمسألة أو إلى إضاعة الوقت بتوليد معلومات غير هامة. تُستخدم هذه النظم في مسائل التفسير حيث نتطلب من النظام أن يخبرنا كل شيء عن معطيات معينة.



النظم المقودة بالمعطيات

استراتيجية النظم المقودة بالهدف تركز على الحل المطلوب. على سبيل المثال، يمكن أن نتطلب من نظام أن يولد خطة لتصنيع منتج معين. جميع الخطط الأخرى غير هامة للمسألة. للوصول إلى الهدف نستنتج الأهداف الجزئية التي توصل إلى هذا الهدف ونحاول تحقيقها.



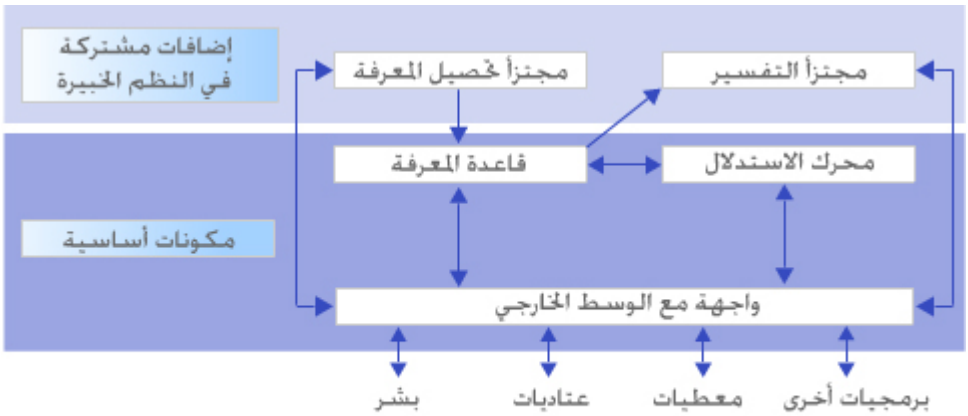
النظم المقودة بالهدف

4. النظم الخبيرة

النظم الخبيرة هي نظم قواعد معرفة، مصممة لضم الخبرة وتجميعها في نطاق معين. من أمثلة نطاقات الخبرة: تشكيل شبكة حواسيب **computer net configuration**، أو تشخيص أعطال الهاتف...الخ. نتطلب من النظام الخبير أن يتصرف كالإنسان الخبير الذي يمكن استشارته في طيف واسع من المسائل ضمن نطاق خبرته.

نموذجياً، يدخل مستثمر النظام الخبير في حوار يصف فيه المسألة (مثل أعراض عطل ما)، ويقدم النظام الخبير النصيحة والاقتراحات أو التوصيات. يمكن أن يقود النظام الخبير الحوار، فيجعل المستثمر يجيب على مجموعة من الأسئلة أو يدخِل معلومات إلى وريقة جدولة **spreadsheets**. يمكن للنظام أيضاً أن يسمح للمستثمر باتخاذ مبادرة وإدخال معلومات دون أن يطلب منه النظام ذلك.

ولأن النظام الخبير هو نظام قواعد-معرفة يتصرف كمستشار خبير، فعليه أن يقدم بعض الإمكانات التي يتصف بها المستشار البشري، وخاصة تبرير إجابته ومحاكمته **reasoning**. وهذا هو الهدف من بناء مجتزأ التفسير **interpretation** في الشكل -1-.



الشكل -1- المكونات الأساسية لنظام معتمد على القواعد

ومع ذلك، فإن أقصى ماتقدمه النظم الخبيرة في هذا المجتزأ، إعطاء أثر الحقائق والقواعد التي جرى استخدامها للوصول إلى النتيجة. وهذا مكافئ لاقتفاء أثر تنفيذ برنامج عادي.

نواة النظام الخبير **expert system shell**، هو نظام خبير بقاعدة معرفة فارغة. وهي تُباع كرزمة برمجية **package**، تُستَخدم في بناء النظم الخبيرة، ويختلف تعقيدها من نواة إلى أخرى.

من حيث المبدأ يجب أن نكون قادرين على شراء نواة نظام خبير، ثم بناء قاعدة المعرفة وإنتاج نظام خبير. ولكن، نظراً للاختلاف الكبير بين نطاقات الخبرة؛ يصعب على مصمم البرمجيات أن يبني نواة نظام خبير تناسب جميع النطاقات. أفضل نوى النظم الخبيرة هي التي تتمتع بمرونة في تمثيل المعرفة وفي تطبيقها. بدون هذه المرونة، نحتاج إلى توليد قواعد وحقائق على نحو نموذج مترابط أو أن نجبر النظام على سلوك معين. ومع ذلك، يبقى الأمر أفضل من استخدام برنامج تقليدي.

5. تحصيل المعرفة

لايمكن الحديث عن تمثيل المعرفة في قاعدة معرفة، قبل تحصيل ومعرفة هذه المعرفة. ثمة ثلاثة مناهج لتحصيل المعرفة الهامة والبارزة في نطاق معين:

- استخراج المعرفة من الخبير بطرح أسئلة عليه.
- الشخص الذي يبني النظام هو نفسه خبير بالنطاق.
- النظام قادر على التعلّم الآلي من أمثلة.

المنهج الأول هو المستخدم عموماً إلا إنه محفوف بالصعاب. نسمي الشخص الذي يستخرج المعرفة من الخبير ويرمزها it **codes** في قاعدة المعرفة مهندس المعرفة **cognitive engineer**. نموذجياً، يجري مهندس المعرفة مقابلات مع واحد أو أكثر من الخبراء في نطاق المسألة، ويحاول أن يجعلهم يعبرون عن معارفهم العميقة بطريقة يستطيع هو فهمها.

يمكن تجنّب مصاعب التواصل بين الخبير ومهندس المعرفة باستخدام المنهج الثاني، حيث نتطلب من خبير النطاق أن يصبح هو مهندس المعرفة، أو نتطلب من مهندس المعرفة أن يصبح خبيراً في النطاق.

أخيراً، في العديد من الحالات، إما أن لا تكون المعرفة متاحة أو لايمكن التعبير عنها صراحةً. في مثل هذه الحالات، يفَضل أن يكون النظام قادراً على بناء قاعدة معرفته بنفسه من مجموعة أمثلة. وتوجد تقنيات كثيرة للتعلّم الآلي.

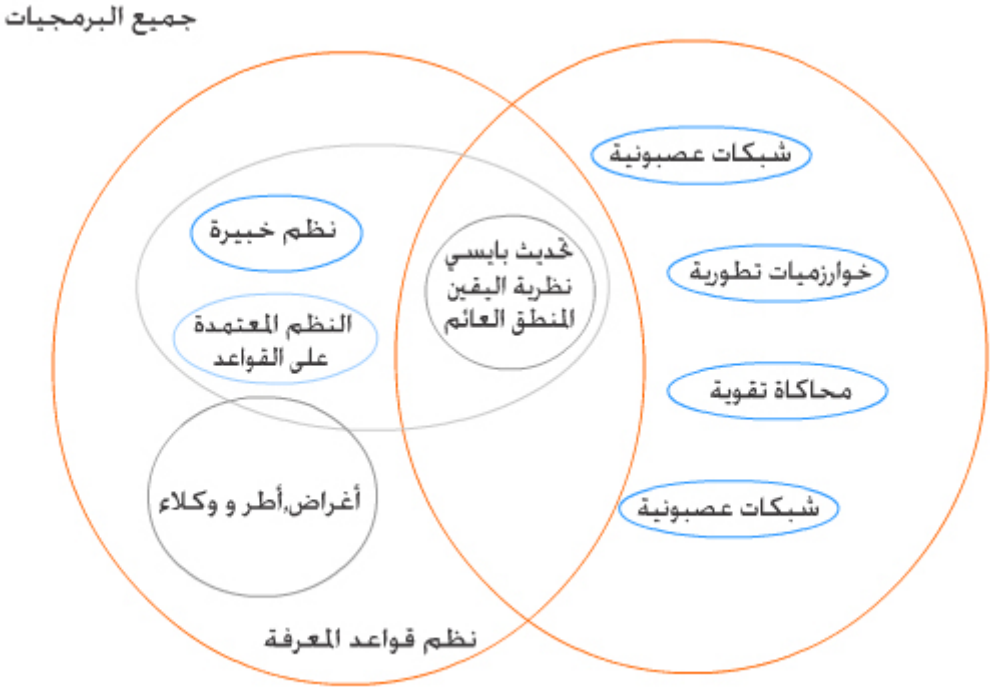
6. الذكاء الحوسبي computational intelligence

لقد تركزت المناقشة على أنواع نظم قواعد المعرفة. تعتمد جميعها التمثيل الرمزي **symbolic representation**، تُمثّل فيها المعرفة صراحةً بكلمات ورموز، نضمها بعضها مع بعض لتشكيل قواعد أوحقائق أوعلاقات أو أي شكل آخر من أشكال تمثيل المعرفة. وبما أن المعرفة مكتوبة صراحة، فيمكن قراءتها وفهمها من البشر. تختلف هذه التقنيات الرمزية عن التقنيات الرقمية المستخدمة في الخوارزميات الجينية والشبكات العصبونية حيث يعبر عن المعرفة بأرقام يضبطها النظام ويحسنها. تُعرف التقنيات الرقمية الذكية بالذكاء الحوسبي **computational intelligence** أو الحوسبة اللينة **soft computing**.

من أهم التقنيات التي تعالج المعارف غير المؤكدة: التحديث البايسي **Bayesian updating**، عوامل اليقين **certainty factors**، والمنطق العائم **fuzzy logic**. تستخدم جميع هذه التقنيات خليطاً من القواعد والقيم الرقمية المرفقة، وبذلك يمكن اعتبارها أدوات للذكاء الحوسبي، وأدوات لنظم قواعد المعرفة بآن.

يبين الشكل -4- أصناف برمجيات النظم الذكية وهي تضم:

- الشبكات العصبونية.
- الخوارزميات الجينية وبعموم أكثر الخوارزميات التطورية **evolutionary algorithms**.
- طرائق احتمالية **probabilistic** مثل التحديث البايسي وعوامل اليقين.
- المنطق العائم **fuzzy logic**.
- تشكيلة من كل ما سبق مع نظم قواعد المعرفة.



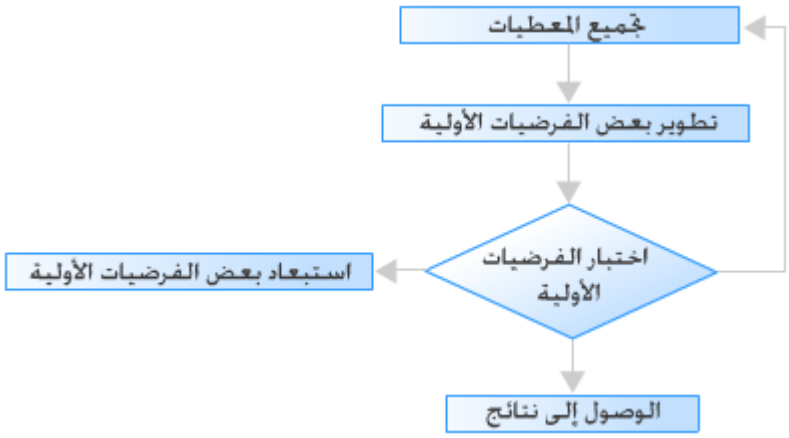
الشكل -4- أصناف برمجيات النظم الذكية

الفصل الثاني

خصائص النظم الخبيرة

1. النظم الخبيرة

في عام **1993** قال **Anderson** أنه يمكن التعبير عن الكثير من أفكار البشر باستخدام قواعد. في الحقيقة، محاكمة البشر ليست عملية سكونية: تُجمع المعطيات، تُطوّر بعض الفرضيات الأولية وتُختَبر، بعضها يستَبعد، وتُطوّر فرضيات أخرى، تُتطلب بعض المعطيات، إلى أن نصل أخيراً إلى بعض النتائج. على البرنامج الحاسوبي -الذي عليه أن يقلد محاكمة البشر- أن يتصرف تصرفاً مشابهاً لما يقوم به البشر لتنفيذ المحاكمة. للأسف يفتقد الوصف الرياضي لإجرائية التفكير هذه الطبيعة الديناميكية.



النظم الخبيرة هي برامج حاسوبية، مصممة لإتاحة بعض مهارات الخبراء للمستثمرين غير الخبراء. ولأن هذه البرامج تحاول أن تحاكي نموذج تفكير الخبير، فمن الطبيعي أن تبرز مثل هذه الأعمال، منذ البداية، في دائرة الذكاء الصنعي. ومن أوائل النظم الخبيرة المعروفة عالمياً نظام **Dendral** الذي يحدد البنى الكيماوية المولية للمركبات من معطيات مقياس طيف الكتلة، والنظام **R1** لتشكيل النظم الحاسوبية، ونظام **MYCIN** للتشخيص الطبي. وتتالت النظم الخبيرة منذ منتصف سيتينيات القرن الماضي وحتى الساعة في مجالات متعددة تتراوح بين عمليات المركبة الفضائية مروراً بنظم تحذير خاصة بمرضى العناية المشددة إلى نظم اتخاذ قرارات مالية.

ثمة طرق عديدة لمقاربة عملية إنشاء برامج حاسوبية تتصرف كالخبير. إحدى أول الطرق المستخدمة هي النظم المعتمدة على قواعد من الشكل **"if...then..."**، لنمذجة إجرائيات محاكمة الخبير (إذا تطابقت المعطيات المتوفرة مع شروط محددة، قُم بفعل مناسب). تتضمن الطرائق الأخرى شبكات الدلالة والشبكات العصبونية والأطر. أما النظم الأفضل (حالياً) فهي التي تستخدم نظام قواعد معرفة مع شبكات عصبونية.

لاتتطلب الشبكات العصبونية أن نحدد نموذج تفكير الخبير صراحةً، وإنما فقط أن تكون لدينا مجموعة من معطيات الدخل مع الخرج الصحيح المناسب لها. يستخدم جزء من المعطيات في التدريب والجزء الآخر في اختبار الشبكة بعد تدريبها.

2. النظم المعتمدة على قواعد المعرفة

تتطلب النظم المعتمدة على القواعد من الخبير أن يذكر صراحة طريقة تفكيره. ويتطلب بناء هذه النظم خبيراً **expert** (أو مجموعة خبراء) في مجال المسألة، ومهندس (أو مهندسي) معرفة متمرساً بالحاسوب. وبعد بناء النظام نتطلب مجموعة معطيات لضبطه تشبه معطيات الاختبار في الشبكة العصبونية ولكنها تكون أصغر حجماً، إضافة إلى مجموعة أخرى لاختباره. إن صعوبة تحصيل المعلومات هي سيئة هذه النظم، وإن كانت النظم العائمة الحديثة تتجاوز هذه الصعوبة بالسماح للخبير بأن يعبر عن خبرته بلغة أقرب إلى المكتوبة من لغة الحواسيب. من جهة أخرى، لنظم قواعد المعرفة إمكان استنتاج قواعد جديدة وإضافة معطيات جديدة على قاعدة المعرفة (مثال نظام **DENDRAL** الذي استنتج مكونات مركبات عضوية).

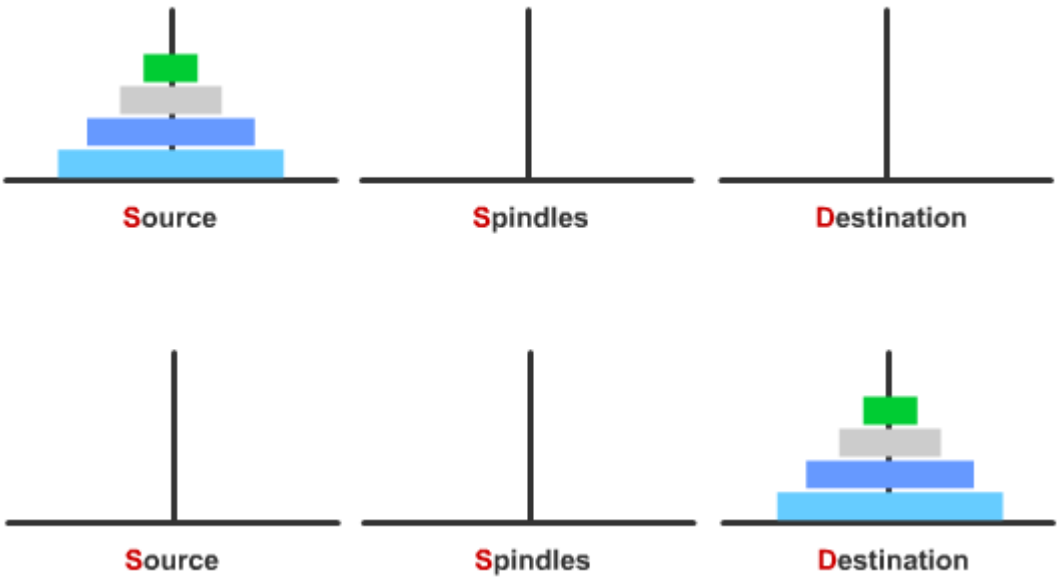
ينظر المتمرسون بالذكاء الصنعي إلى النظم الخبيرة، في غالب الأحيان، على أنها تشبه الأطفال المتخلفين. فقد لا يكون لدى النظم الخبيرة العادية إمكان محاكاة تفكير البشر بشكل كاف، ويعود السبب في ذلك إلى: عدم كفاية إمكاناتها في محاكاة نماذج التفكير المعقدة، ونقص في إمكانات فهم اللغات الطبيعية. تتصدى لغة البرمجة **FLOPS**، على سبيل المثال، للمسألة الأولى حيث تسمح بمحاكاة نماذج تفكير معقدة، إضافة إلى نمطين من التعلم.

ثمة نوع حديث نسبياً من نظم قواعد المعرفة هي النظم العائمة التي تعتمد على القواعد العائمة. ثمة ثلاثة مفاهيم جديدة لبرمجة مثل هذه النظم: **نظرية النظم العائمة والبرمجة غير الإجرائية non procedural المقودة بالمعطيات والبرمجة المتوازية**.

- تسمح نظرية النظم العائمة بمعالجة حالات الشك والغموض والتناقض.
- أما البرمجة غير الإجرائية المقودة بالمعطيات فتعني أن ترتيب تنفيذ القواعد لاعلاقة له بترتيب ورود هذه القواعد في البرنامج وإنما بترتيب ورود المعطيات.
- أما البرمجة المتوازية فتعني أن جميع القواعد القابلة للتطبيق تُنفذ دفعة واحدة.

من وجهة نظر برمجية، إذا كان النظام الخبير مكتوباً بلغة عادية مثل **C++** فنقول إنها نظام إجرائية؛ أما إذا استخدمت قواعد من الشكل **if...then...** فنقول إنها نظام إنتاج **production systems** أو نظم قواعد معرفة.

مسألة "برج هانوي" هي مسألة نموذجية من مسائل الألعاب في الذكاء الصنعي. في هذه المسألة، نفترض لدينا ثلاثة قضبان، وعدداً من الأقراص المتناقصة الأقطار من الأسفل إلى الأعلى، على أحد القضبان. والمسألة المطلوب حلها هي نقل هذه الأقراص من قضيب إلى آخر؛ ننقل قرصاً في كل حركة، ونحرص على ألا نضع قرصاً على قرص آخر أصغر منه قطراً.



يمكن وضع قاعدة إنتاج لحل هذه المسألة من الشكل:

قاعدة:

إذا (كان لدينا قرص واحد يمكن تحريكه وكان موجوداً في **S** ونود نقله إلى **D**) حرك القرص من **S** إلى **D**.

```
rule (goal Fires if only one disk to move)
IF (in Spindles n = 1 AND source = <S> AND destination = <D>) THEN
write "*** move <S> to <D> ***\n",
delete 1;
```

في القاعدة السابقة، القضيب **spindles** هو بنية معطيات تتضمن **n** عنصراً (عدد الأقراص على القضيب)، والمصدر **S Source** هو القضيب الذي ننقل منه الأقراص، والوجهة **Destination**

D هو القضيب الذي ننقل إليه الأقراص. بعد تنفيذ القاعدة نكتب رسالة على الشاشة، ونحذف منتسخات **instances** القضبان.

3. النظم المقودة بالمعطيات

تُعتبر النظم المقودة بالمعطيات ذات أهمية خاصة. وهي تستخدم لغات برمجة تختلف عن اللغات الإجرائية العادية مثل **C** أو باسكال **Pascal** أو فورتران **Fortran** أوبيسيك **Basic**. ففي اللغات الإجرائية، تُنفذ التعليمات التي تشكل الجزء الأكبر من اللغة، بالترتيب الذي كُتبت فيه (إلا إذا حصل نقل للتحكم ووردت تعليمات تحكم مثل تنفيذ برنامج جزئي أو غيره)؛ في اللغة المقودة بالمعطيات، القواعد التي تشكل جزءاً كبيراً من اللغة مرشحة للتنفيذ حالما تحققت المعطيات في ترويسة (شرط) القاعدة **IF part**.

إذا حققت المعطيات أكثر من قاعدة بآن، تقوم لغات البرمجة العامة المعتمدة على القواعد مثل **OPS** بتنفيذ القواعد تسلسلياً، واحدة في كل مرة. علينا أولاً أن نحدد القواعد التي أصبحت قابلة للقدح نتيجة تحقق المعطيات في ترويستها. ثم تقرر خوارزمية حل التصادمات **rule conflict algorithm** أي هذه القواعد يجب قدحها (تنفيذها). توضع القواعد القابلة للقدح، التي لم يجر قدحها بعد، في مكدس تمهيداً لقدحها، إذا لم تنشأ قواعد جديدة قابلة للقدح نتيجة تطبيق القاعدة المنفذة مؤخراً يجري قدح القاعدة المختارة، أي تنفيذ الجزء **THEN** منها، ثم نعود ثانية للبحث عن القواعد الأخرى القابلة للقدح.

4. ميزات خاصة بالنظم العائمة

قد لا تكون لدى النظم الخبيرة التقليدية الإمكانات الكافية لمحاكاة تفكير البشر محاكاة جيدة وذلك لسببين:

- عدم القدرة على محاكاة نماذج التفكير المعقدة.
- نقص في إمكانات فهم اللغات الطبيعية.

ثمة لغة برمجة **FLOPS** تحل المشكلة الأولى دون الثانية، اعتماداً على لغة صورية **formal language**.

تعتمد النظم الخبيرة العائمة على ثلاثة مفاهيم جديدة: **نظرية النظم العائمة، والبرمجة غير الإجرائية المقودة بالمعطيات والبرمجة المتوازية**.

- تسمح نظرية النظم العائمة بمعالجة الشك والغموض والتناقض.
- البرمجة غير الإجرائية المقودة بالمعطيات تعني أنه حين تكون لغة **FLOPS** بنمط التنفيذ، وتقدح قواعد (تنفذها)، فإن ترتيب تنفيذ القواعد لا علاقة له بترتيب ظهور هذه القواعد في البرنامج وإنما فقط بالمعطيات المتاحة التي تجعل قاعدة أو مجموعة قواعد مؤهلة للقدح أو معطلة.
- أما لغة البرمجة المتوازية فتعني أن كل القواعد القابلة للقدح تُنفذ معاً، عوضاً عن تنفيذها بترتيب مسبق.

تتميز النظم الخبيرة العائمة بأن معظمها يوفر إمكان القدح المتوازي لجميع القواعد القابلة للقدح: أي إن جميع القواعد القابلة للقدح، تُقدح فعلياً في الوقت نفسه، محاكين بذلك الحواسيب المتوازية.

لعملية التوازي هذه فوائد عديدة في النظم العائمة. وللغات المتوازية أهمية خاصة حين نتعامل مع المجموعات العائمة:

- تجعل البرمجة أسهل.
- تجعل التنفيذ أسرع.

وذلك مقارنة بالنظم المكافئة التي تُنفذ تسلسلياً. ومع ذلك، فللبرمجة التسلسلية فوائدها أيضاً في بعض الحالات؛ فهي مناسبة حين نود استنباط معلومات من المستثمر حين يتعلق كل سؤال نود

طرحه بجواب السؤال السابق. وبناء على ذلك، يجب أن تدعم لغات برمجة النظم الخبيرة العائمة إمكان قدح القواعد بالنمط التسلسلي وبالنمط المتوازي. هذه السمة غير مألوفة لمعظم المبرمجين.

وعلى من يود بناء نظام خبير عائم أن يتعلم النقاط التالية:

- العمل مع القواعد الشرطية **IF-THEN** كجزء أولي من اللغة.
- تعلم أساسيات نظرية النظم العائمة: المجموعات العائمة والمنطق العائم والأعداد العائمة.
- تعلم لغات البرمجة غير الإجرائية المقودة بالمعطيات.
- التعامل مع تنفيذ لغات البرمجة بالنمط التسلسلي وبالنمط المتوازي.

5. نظم خبيرة للتحكم والمحاكمة

ثمة نمطان من النظم الخبيرة العائمة: **نظم التحكم العائم ونظم المحاكمة العائمة**، ومع أن كلاهما يستخدم المجموعات العائمة، إلا أنهما يختلفان بالمنهجية.

لقد أُنجِز أول نظام تحكم عائم ناجح في عام **1976** بهدف التحكم بمصنع إسمنت. ومنذ ذلك الحين، لاقت النظم العائمة رواجاً في تطبيقات التحكم في اليابان ثم في العالم أجمع.

يمكن توصيف **نظام التحكم العائم** ببساطة كما يلي: هو نظام يتقبل دخولاً **inputs** على هيئة أرقام، ثم يترجمها إلى كلمات لغوية مثل: بطيء، متوسط، وسريع، الخ. وهذا ما يسمى التعويم **fuzzification**. بعد ذلك، تقوم القواعد بمطابقة هذه الدخول اللغوية لتعطي مفردات لغوية مشابهة تصف الخروج **outputs**. أخيراً تجري ترجمة مفردات الخروج اللغوية إلى خروج على هيئة أرقام، وهذا مايسمى إزالة التعويم أو فك التعويم **defuzzification**. إن النحو الذي تُكتب فيه القواعد في هذه الحالة مناسب لأهداف التحكم، إلا أنه قاصر عن المحاكمة العائمة. توجد بيئات متاحة لبناء نظم التحكم العائم. نعطي فيما يلي مثالاً على قاعدة تحكم عائمة:

إذا كان الدخل الأول عالياً، والدخل الثاني منخفضاً، يكون الخرج معدوماً"

لايمكن وصف قواعد المحاكمة العائمة بهذا الشكل المتراص. إن مجال تطبيقات نظم التحكم العائمة معروفٌ تماماً؛ وهي تعمل بشكل جيد لدخول وخروج رقمية على هيئة أعداد. أما مجال نظم المحاكمة العائمة فهو غير معرف؛ إذ نعرف نظم المحاكمة العائمة كما يلي: تحاول نظم

المحاكمة العائمة أن تحاكي التفكير البشري، لذلك فهي لا تمتلك قيوداً أولية على ذلك التفكير. وفي حين أن نظم التحكم العائمة تتعامل مع أرقام، فإن نظم المحاكمة العائمة تتعامل مع معطيات رقمية وغير رقمية. على سبيل المثال، دخل نظام المحاكمة العائمة الذي يتعلق بدرجات الحرارة يمكن أن يكون "**38** درجة مئوية" أو "حرارة ضعيفة" ويمكن أن يكون الخرج: "وافق" أو "انقل تجهيزة" أو "شغل مكيف". بالمقابل، لاتحتاج القواعد هنا إلى عملية تعويم أو إزالة تعويم، وإنما يمكن تقسيمها إلى عمليات منفصلة يمكن أن تُنجز أو لا تُنجز بحسب متطلبات المسألة.

يتقبل نَحو **syntax** قواعد نظم المحاكمة العائمة تشكيلة واسعة من الأنماط، نذكر منها على سبيل المثال النمطين التاليين:

1. "إذا كانت لدينا أعراض اكتئاب، لمدة أكبر من ست سنوات فالتشخيص اكتئاب شديد"، مع أن هذه القاعدة تشبه قواعد التحكم العائم إلا أنها مختلفة عنها تماماً. في قاعدة التحكم العائم يجب أن يكون "العرض" رقمياً، هنا "العرض" مجموعة عائمة من المفردات اللغوية أحد عناصرها "الاكتئاب". كذلك، "التشخيص" يجب أن يكون عددياً في قاعدة التحكم العائم أما في قاعدة المحاكمة العائمة فهو مجموعة عائمة من التشخيصات أحد عناصرها "الاكتئاب".
2. يمكن أن يولد الهدف قاعدة جديدة، استجابةً لإثارة معينة. كما في المثال التالي، حيث لدينا محسات نقرأ منها قيماً من محسات العداد **id2** ,**id1** وفي التوصيلات **id** لتجهيزة معينة، ونقرأ الجواب، فإذا طابقت القراءات تسلسل معطيات معين، طبق القاعدة **3**:

"إذ كان (في العداد **id1=<id2>** و **id2=<id1>** وفي التوصيلات **id=<id2>** والجواب **<R>** عندها:

طبق القاعدة **3** وإذا كانت (في المرحلة اللاحقة **id=<id2>** عندها أرسل رسالة)"

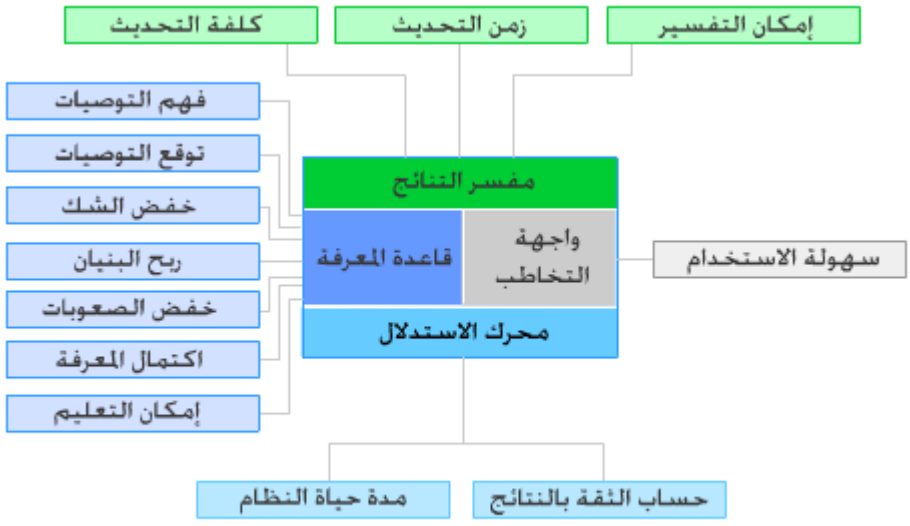
هذه القاعدة لا تشبه إطلاقاً قواعد التحكم العائم. وفي ظروف محددة، يمكن إنشاء قواعد جديدة.

6. خواص النظم الخبيرة الناجحة

لقد تطور استخدام النظم الخبيرة بهدف مساعدة الشركات الحديثة على تحسين نوعية خدماتها وأدائها المالي وحل مشاكلها العملية المعقدة **complex business problems**. وفي أحد الأبحاث في اليونان، عرف باحثون الخصائص التي يجب أن تتمتع بها النظم الخبيرة لاعتبارها ناجحة وتؤدي الغرض من بنائها، معتمدين بذلك على التحليل العملي لثلاثة نظم خبيرة أثبتت جدارتها في الصناعة، وقد اقترحوا أربع عشرة فرضية يجب أن يحققها النظام الخبير الناجح، ودرسوا مدى تحققها في النظم المدروسة الثلاثة.

مع أن الكثير من الشركات المتوسطة والكبيرة تتبنى تقانات النظم الخبيرة كأداة لتحسين الإنتاج والتنافسية إلا أنه لازال الكثير من الشكوك يدور حول أدائها بسبب التوقعات الهائلة التي كانت تُنتَظر من التطبيقات الأولية للنظم الخبيرة، وقصر زمن حياة تلك النظم. تتحدث المراجع عن نسبة فشل من مرتبة **90%** للنظم الخبيرة لأسباب متعددة: قِصر في النمذجة، ضعف في محرك الاستدلال، نقص وضعف في المعارف، وغيرها. إضافة إلى ذلك ثلث النظم الخبيرة الناجحة فقط بقيت أكثر من عشر سنوات وقد عزا الباحثون ذلك لأسباب فردية ومؤسساتية وليس تقنية.

في عام **1990** جرت دراسة حول النظم الخبيرة، عرفت مجموعة من النقاط لأخذها بالاعتبار حين تقويم النظم الخبيرة، منها: سهولة الاستخدام، ودقة التوصيات الناتجة عنها، إمكان حل المسائل من دون خلق مسائل جديدة. وفي عام **1995** جرت دراسة أخرى بينت أن من عوامل نجاح النظم الخبيرة: إمكان دعم إدارة النظام، وحل مسائل فورية، ودرجة التعديل المنظم للنظام **regular modification** ومشاركة المستخدم في بناء النظام. يبين الشكل **-1-** خواص النظام الخبير الناجح بحسب دراسة أُجريت في **1999**.



الشكل-1- مقومات نجاح النظام الخبير بحسب دراسة أجريت في عام **1999**

الخواص التي سنتحدث عنها هنا مستقاة من نظم خبيرة أثبتت جدارتها، ولازالت قيد الاستخدام منذ فترة طويلة، وقد شارك الباحثون أنفسهم في بنائها (انظر المرجع). يبين الجدول التالي هذه النظم التي جرى تطويرها في مخبر نظم دعم اتخاذ القرار **Decision Support System** في الجامعة الوطنية التقنية في أثينا.

	نطاق الخبرة	مدة الحياة	منصة التطوير	الحالة الراهنة
GENESYS	جدولة الإنتاج	2000- today	NEXPERT OBJECT, SQL Server, Ms Windows	مستخدم في الإنتاج في شركة أدوية يونانية
RECOT	تشخيص في الصناعات النسيجية	1999- today	GURU, C++, Ms Windows	مستخدم في الإنتاج في شركة نسيج يونانية
MOAS	ERP تخطيط موارد الشركات نمذجة وتكييف	2001- today	XpertRule, ORACLE, DELPHI	مستخدم كأداة دراسة حالة لمزود ERP في اليونان Software House in Greece

الجدول **-1-** ثلاثة نظم خبيرة طُورت في مخبر نظم دعم اتخاذ القرار في الجامعة الوطنية التقنية في أثينا

1. النظام الأول **GENESYS (GENeric Expert System for Schedualing)** لحل مشكلة الجدولة. يأتي تعقيد مسألة الجدولة من وجود شكوك ديناميكية على العملية: تعطل آلات، طلبات مفاجئة، تغير موعد التسليم، الخ. ويمكن تعريف الجدولة على أنها عملية تخصيص موارد **resources allocation** محدودة إلى مهام عبر الزمن، للحصول على الخرج المرجو في الوقت المناسب، وبحيث توجد قيود زمنية وعلاقاتية بين المهام والموارد. تتضمن قاعدة معرفة النظام الخواص المتعددة للآلات والمحيط إضافة إلى أهداف الإنتاج. يضع المستثمر شروطه ويقترح النظام أفضل الطرق لحل مسألة الجدولة **scheduling** اعتماداً على قواعد **if-then** أو خوارزمية جدولة. يستخدم النظام نواة **shell** هي **NEXPERT OBJECT** من شركة **Neuron Data** تسمح هذه النواة ببرمجة قواعد. يتضمن النظام **400** قاعدة، وله إمكان التعامل مع رسائل خارجية.
2. النظام الثاني **RECOT Reduction of the Environmental Cost in the Textile industry**. تُعتبر صناعة النسيج من الصناعات المتطورة في أوروبة، والتي تلعب الخبرة فيها دوراً كبيراً سيما في مجال تخفيض تلوث محيط المصنع بحسب المواصفات

الأوروبية، ولا يمكن تحليل أثر التلوث قبل صباغة النسيج. عدا عن ذلك، هناك ضرورة لأمثلة **optimization** استهلاك الماء والطاقة. لقد جرى تصميم النظام **RECOT** للإجابة على هذه المتطلبات، وقد جرى تطويره في بيئة **GURU** للنظم الخبيرة وهو يعمل الآن تحت نظام **Windows NT** ويدعم التخاطب مع **ODBC** (قواعد معطيـات) و **APIs** في عدة لغات برمجة: لغة **C++** و **Visual Basic**. تتضمن قاعدة معرفة النظام **50** صفاً وصفاً جزئياً، و**120** غرضاً و **430** قاعدة.

3. النظام الثالث **MOAS Modeling and Optimal Adaptation System** لنظام تخطيط موارد الشركات **ERP**. يسعى النظام إلى نمذجة خبرة المستشارين في مجال **ERP** حول نمذجة الشركات، إضافة إلى معارف تقنية لتكيف نظام **ERP** مع إجراءات العمل في الشركة. جرى تطوير النظام في بيئة **XpertRule** وتتضمن قاعدة المعرفة فيه **500** قاعدة.

بعد دراسة هذه النظم الثلاث، تبين وجود **14** فرضية لنجاح النظم الخبيرة، نبينها في الجدول **-2-** مع مدى تحقيق كل من النظم المدروسة لهذه الفرضيات:

	الفرضية	مدى تحقيقها	GENESYS	RECOT	MOAS
مجال التطبيق	ضيق المجال	نعم	+	+	+
	إمكان عالٍ للأتمتة	مختلط	+	-	+
	درجة عالية من التكرار	نعم	+	+	+
	صغر المشروع	نعم	+	+	+
التطوير	النظام يدعم الخبير ولا يستعيض عنه	نعم	+	+	+
	قيمة مضافة للمستثمر	نعم	+	+	+
	فوائد مبكرة للخبير	نعم	+	+	+
	الحلول المباشرة البسيطة هي الأفضل	نعم	+	+	+
	سرعة التطوير	نعم	+	+	+
	إدارة الخطأ (الانزياح)	نعم	+	+	+
	استخدام نواة نظام خبير	نعم	+	+	+
	النظام الخبير جزء من نظام متكامل	نعم	+	+	+
	إمكان تطبيق قواعد تطوير البرمجيات	نعم	+	+	+
العمليات	توجه نحو البرمجيات والعتاديات العامة	نعم	+	+	+

الجدول-2- : خواص النظام الخبير الجيد بحسب فرضيات دراسة الباحثين

الفصل الثالث

النظم الخبيرة المعتمدة على القواعد

1. الحقائق والقواعد

إن النظم الخبيرة المعتمدة على القواعد **rule-based expert systems** هي نظم معرفة **knowledge base system**. تتضمن قاعدة المعرفة فيها مجموعة (أو مجموعات) من القواعد. القواعد **rules** هي طريقة صريحة وبسيطة وأنيقة ومباشرة للتعبير عن المعرفة، وأبسط نمط من القواعد هو ما يعرف بقواعد الإنتاج **production rules** وهي من النمط:

إذا <تحقق شرط> إذن <نحصل على نتيجة> (**If <condition> then <conclusion>**).

لدينا القاعدة التالية، على سبيل المثال:

إذا <كان الصنبور مفتوحاً> إذن <يتدفق الماء منه>

ما يميز القواعد هو أنها قريبة جداً من اللغة الطبيعية، فالقاعدة السابقة يفهمها كل من يتكلم العربية. والجدير بالذكر أن بالإمكان صياغة قواعد أعقد من هذه القاعدة بكثير.

لكي نتمكن من تطبيق القواعد وبالتالي استخدام نظام القواعد، يجب أن يكون النظام قادراً على النفاذ إلى الحقائق.

الحقائق **facts** هي عبارات غير شرطية، ويفتَرض أن تكون صحيحة لحظة استخدامها. <الصنبور مفتوح> هي حقيقة، ولكن من أين نحصل على الحقائق؟ ثمة عدة خيارات للحصول على هذه الحقائق:

- الحقائق موجودة في قاعدة معطيات.
- الحقائق مخزنة في ذاكرة الحاسوب.
- الحقائق يحددها مجموعة محسات مرتبطة بالحاسوب.
- الحقائق يمكن طلبها من المستثمر.
- الحقائق يمكن اشتقاقها من قواعد طُبّقت سابقاً نتيجة حقائق سابقة.

يمكن النظر إلى الحقائق على أنها قواعد خاصة، جزء الشرط فيها صحيح دائماً. فالحقيقة <الصنبور مفتوح> يمكن النظر إليها على أنها القاعدة التالية:

إذا <شرط صحيح دائماً **true** > إذن <الصنبور مفتوح>.

وبأخذ هذه القاعدة، التي تمثل حقيقة أن <الصنبور مفتوح>، مع القاعدة:

إذا <كان الصنبور مفتوحاً> إذن <يتدفق الماء منه>

يمكننا استنتاج حقيقة جديدة وهي أن <يتدفق الماء من الصنبور>. هذه الحقيقة الأخيرة تُخزن في ذاكرة الحاسوب، ويمكن أن تحقق جزء الشرط من قواعد جديدة، وتؤدي لاشتقاق حقائق جديدة. مجموعة الحقائق المعروفة للنظام في لحظة معينة تشكل قاعدة الحقائق **fact base**.

تتطلب كتابة القواعد لغة برمجة تصريحية **declarative**، وتمثل هذه القواعد المعارف التي يمكن أن يستخدمها الحاسوب، دون الإشارة إلى كيفية وزمن تطبيق هذه القواعد. من حيث المبدأ، يجب ألا يكون ترتيب كتابة القواعد في البرنامج ذا أهمية، وأن يكون بالإمكان إضافة قواعد جديدة في أي لحظة. ولكن الأمر ليس دائماً هكذا..

نتطلب محرك استدلال **inference engine** ليفسر القواعد ويطبقها عند اللزوم. توجد هذه المحركات في مجموعة من أدوات البرمجة مثل: نوى (ج. نواة) النظم الخبيرة **expert systems' shells**، أدوات الذكاء الصنعي، مكتبات برمجية ولغة برولوغ **Prolog**، وغيرها.

مثال: في نظام خبير للتحكم بغلاية ماء يمكن أن يكون لدينا مجموعة قواعد مثل:

- إذا كان مستوى الماء منخفضاً، افتح صمام التحكم.
- إذا كانت الحرارة عالية ومستوى الماء منخفضاً، افتح صمام التحكم وأغلق أنابيب الغلاية.
- إذا كان مخرج البخار مسدوداً، بدّل أنبوب الخرج.
- ...
- إذا كانت الحرارة عالية ولم يكن مستوى الماء منخفضاً إذن الضغط عالٍ.

القواعد الأولى تمثل معرفة سطحية والقاعدة الأخيرة تمثل معرفة عميقة لها علاقة بقوانين الفيزياء. الحقائق نحصل عليها من محسات الحرارة والضغط وغيرها لمعرفة إذا كانت القواعد قابلة للتطبيق.

أما تطبيق القواعد فيتولاه محرك الاستدلال بحسب الخطوات التالية:

- اختيار القواعد لفحصها: نستنتج القواعد المتاحة **available rules**.
- تحديد أي القواعد قابلة للتطبيق، هذه القواعد هي القابلة للقدح **triggered rules**، وهي تشكل مجموعة التعارض.
- اختيار القواعد التي ستُقدح **to fire**.

هذه الطريقة تسمى السلسلة الأمامية، إذ ننظر في القواعد التي يكون جزء الشرط فيها محققاً

فنعتبرها قابلة للتطبيق.

2. السلسَلة الأمامية و السلسَلة الخلفية

في السلسلة الأمامية **forward chaining**، نتفحص قاعدة الأفعال، ثم نسرد القواعد القابلة للتطبيق لأن الأفعال في جزء الشرط منها محققة، ثم نطبق هذه القواعد ونستنتج أفعالاً أخرى ونعيد الكرة. نقول إن هذه الاستراتيجية مقودة بالمعطيات **data-driven strategy**، ويمكن في أي لحظة إضافة قواعد جديدة. ولكن، إذا كانت لدينا مجموعة قواعد قابلة للتطبيق بنفس اللحظة فأي القواعد نبدأ بها؟ وإذا أضفنا قاعدة جديدة تتعارض مع قاعدة سابقة فأيها يجب أن يؤخذ بالاعتبار؟ كل هذا نراه في فقرة لاحقة هي حل التعارض.

من جهة أخرى، إذا لم تكن الحقائق **facts** صحيحة **true** فإننا نفترضها خاطئة (غير محققة) **false**، يعرف هذا الافتراض بفرضية العالم المغلق. لو لم نقم بهذا الافتراض لوجب إضافة حالة ثالثة للحقيقة **fact** مثلاً، كأن تكون هذه الحقيقة غير معروفة **unknown**.

يمكن أن تتضمن القواعد متغيرات لجعلها أكثر تراصاً **compact more**. فمثلاً في حالة الغلاية **boiler**، إذا كان لدينا مجموعة القواعد التالية:

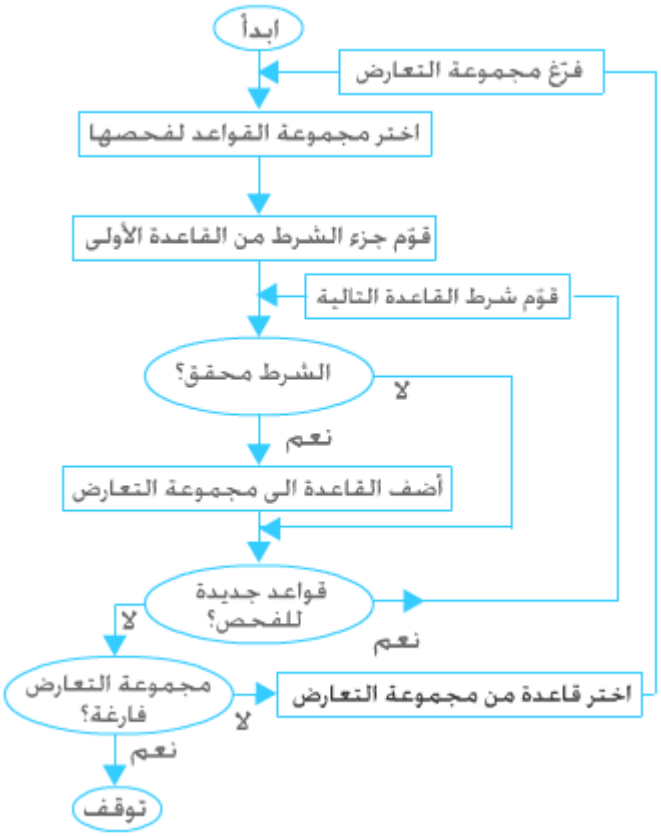
- إذا كان صمام التحكم **2** مفتوحاً كان معدل التدفق في الأنبوب **2** عالياً.
- إذا كان صمام التحكم **3** مفتوحاً كان معدل التدفق في الأنبوب **3** عالياً.
- إذا كان صمام التحكم **4** مفتوحاً كان معدل التدفق في الأنبوب **4** عالياً.
- ...

عندها يمكن الاستعاضة عن مجموع هذه القواعد بقاعدة تتضمن متغيراً من الشكل:

- إذا كان صمام التحكم **x?** مفتوحاً كان معدل التدفق في الأنبوب **x?** عالياً.

غالباً نشير إلى اسم المتغير بجعله مسبوقاً بإشارة الاستفهام **?**. حين يأخذ المتغير القيمة **2** مثلاً نقول أننا أخذنا منتسخاً **instance** من المتغير، ونقول أن المتغير قد توحّد **unified** مع ثابت المنتسخ.

يبيّن الشكل -1- خوارزمية السلسلة الأمامية.



الشكل -1- خوارزمية السلسلة الأمامية

بالمقابل السلسلة الخلفية **backward chaining** هي استراتيجية استدلال تفترض وجود هدف نود تحقيقه أو دحضه. مثلاً في حالة نظام الغلاية السابق، نود أن نعرف إذا كان علينا تبديل أنبوب خرج الغلاية أم لا، بقطع النظر عن جميع الاستدلالات الأخرى التي يمكن للنظام أن يستنتجها.

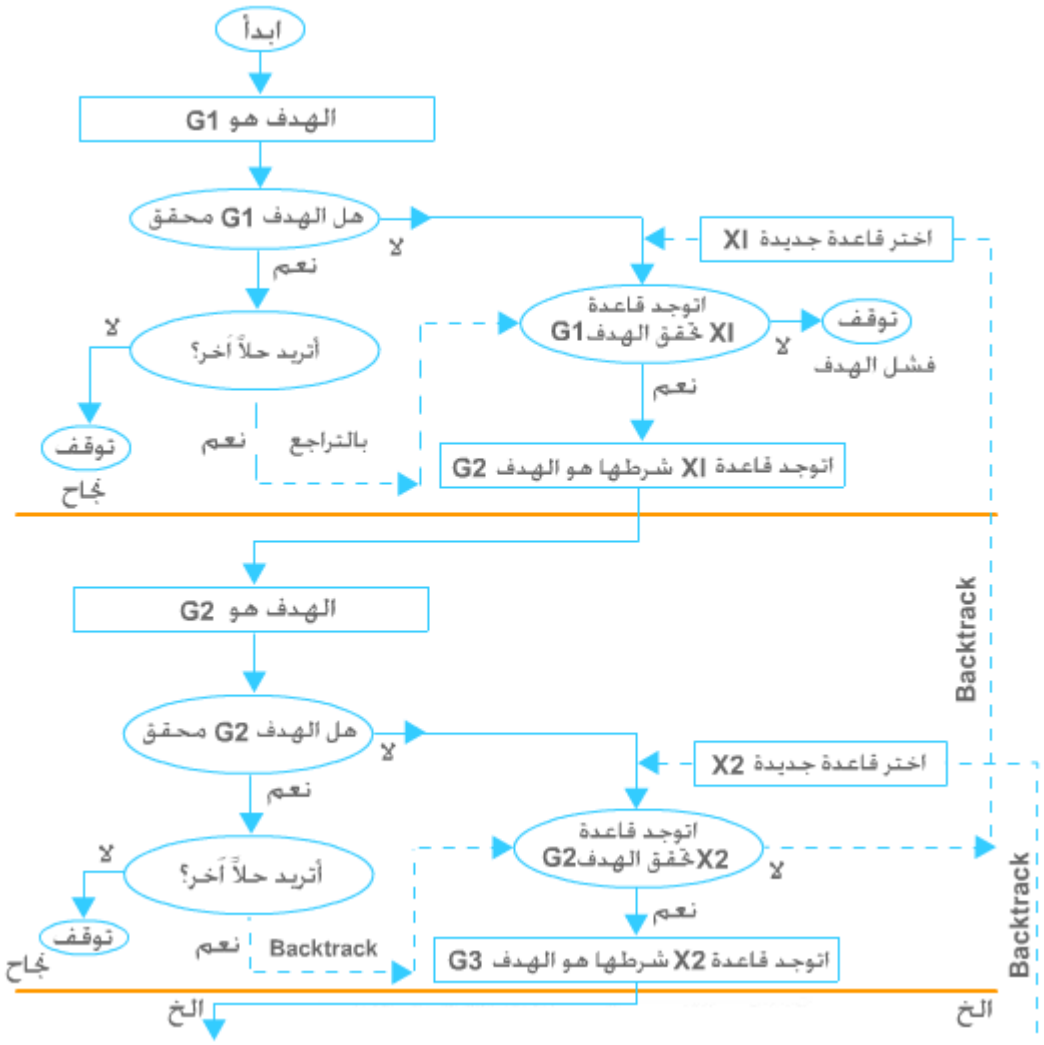
نبدأ باختيار القواعد التي تقود مباشرة إلى تحقيق هذا الهدف. في مثالنا القاعدة هي القاعدة الثالثة:

- إذا كان مخرج البخار مسدوداً، بدّل أنبوب الخرج.

ولكن ليس لدينا في قاعدة الحقائق ما يدل على أن مخرج البخار مسدوداً، لذلك لا يمكن تنفيذ هذه القاعدة، وعوضاً عن ذلك نبحث عن هدف جديد هو التحقق من أن مخرج البخار مسدود، وهكذا... إلى أن نصل إلى حقائق موجودة في قاعدة الحقائق تحقق الهدف الجزئي، وبـالتراجع إلى تحقيق الهدف الأصلي.

في استنتاج الأهداف الجزئية يمكن أن نتحرك عمقاً-أولاً (وهي الاستراتيجية التي شرحناها، وتطبق في غالب الحيان) أو عرضاً-أولاً بالنظر في كل القواعد التي تحقق الهدف الجزئي.

يبين الشكل -2- مخطط السلسلة الخلفية.



الشكل -2- خوارزمية السلّسلة الخلفية

لغة برولوغ **Prolog** هي لغة تصريحية تعمل بالسلسلة الخلفية أو بالتراجع **backtracking**، يحصل التراجع في حالتين:

- حين لا يتحقق الهدف في مجموعة القواعد قيد الاعتبار.
- حين يتحقق الهدف ولكن يرغب المستثمر في حل آخر أو في طريقة أخرى لتحقيق الهدف.

تعرف الخوارزمية التالية طريقة لتنفيذ السلسلة الخلفية باستخدام العودية **recursion**.

```
define function backwardchain(G);
        /* returns a boolean (i.e., true/false) value */
        /* G is the goal being validated */
variable S, X, C;
        result:= false;
        /* ':=' represents assignment of a value to a variable */
        S:= set of rules whose conclusion part matches goal G;
        if S is empty then
                result:= false;
        else
                while (result=false) and (S is not empty) do
                        X:= rule selected from S;
                        S:= S with X removed;
                        C:= condition part of X;
                        if C is true then
                                result:=true
                        elseif C is false then
                                result:=false
                        elseif (backwardchain(C)=true) then
                                result:=true;
                        /* note the recursive call of 'backwardchain' */
                        /* C is the new goal */
                        endif;
                endwhile;
        endif;
return result;
        /* 'result' is the value returned by the function */
        /* 'backwardchain' */
enddefine;
```

الشكل **-3-** خوارزمية عودية للسلسلة الخلفية

وثمة عدة متغيرات لتنفيذ السلسلة الخلفية بحسب الرغبة:

- هل البحث في القواعد عرضاً أولاً أم عمقاً أولاً؟
- هل نريد تَتَبع الحلول الأخرى؟
- كيف نختار بين الفروع (في البحث عمقاً أولاً)؟
- كيف نختار ترتيب القواعد التي نفحصها (في البحث عرضاً أولاً)؟
- حين نصل إلى تتالي القواعد الذي يحقق الهدف، هل ننفذ تتالي القواعد أم نستنتج فقط أن الهدف محقق؟

3. حل التعارض

ثمة عدة طرق لحل التعارض لاختيار القاعدة التي نبدأ بتنفيذها من مجموعة القواعد القابلة للتنفيذ، والتي ندعوها مجموعة التعارض **conflict set**، نذكر منها:

القادم أولاً يُخدم أولاً

تنفيذ أول قاعدة جرى اختيارها لتكون في مجموعة التعارض. وهذا يعتمد على ترتيب كتابة القواعد في مجموعة القواعد.

بحسب الأولوية

عوضاً عن الاعتماد على ترتيب كتابة القواعد، نقوم بإسناد رقم أولوية لكل قاعدة حين نكتبها. وحين يكون لدينا مجموعة قواعد في مجموعة التعارض نبدأ بتنفيذ القاعدة ذات الأولوية الأعلى وهكذا.

قواعد مترفعة metarules

القواعد المترفعة هي قواعد لاتتعلق بالمعرفة حول التطبيق الذي نحن بصدده، ولكن بالمعرفة حول كيفية تطبيق القواعد. لذلك، نسمي هذه القواعد القواعد المترفعة "قواعد على القواعد". في مثالنا السابق، يمكن أن تكون لدينا القاعدة المترفعة التالية:

نفضل القواعد التي تتعلق بإطفاء النظام على القواعد التي تتعلق بالتحكم في الصمامات.

4. معالجة الشك

إن ما ناقشناه حول نظم قواعد المعرفة يفترض أن العالم حولنا واضح، وأن جميع فرضياتنا صحيحة أو خطأ أو غير معروفة. وأحياناً نفترض أن أي فرضية غير معروفة هي فرضية خطأ. ولكن العالم الواقعي نادراً ما يكون بهذا الوضوح.

لتكن لدينا القاعدة التالية، على سبيل المثال:

إذا كان خرج المحول منخفضاً كان مستوى الماء منخفضاً

تتضمن هذه القاعدة ثلاثة أشكال من الشك **uncertainty**:

- شك في القاعدة نفسها.

فمستوى الماء المنخفض ليس التفسير الوحيد لخرج المحول المنخفض. وما يريد الخبير قوله من هذه القاعدة، أنه إذا كان خرج المحول منخفضاً **فمن المحتمل** أن يكون

مستوى الماء منخفضاً.

- شك في الدليل:

قد لايكون الدليل الذي تعتمد عليه القاعدة مؤكداً. وقد يكون هناك سببان لهذا الشك: فقد يأتي الشك من مصدر غير موثوق، كأن يكون جهاز قياس المستوى لا يعطي نتائج دقيقة، وقد يكون الدليل نفسه آتٍ من قاعدة محتملة وليست مؤكدة.

- استخدام لغة غامضة:

لا يمكن تكمية كلمات مثل: منخفض، عالٍ، كثير، قليل،...فإذا قلنا مستوى المحول منخفض لا ندري هل هو **1mV** أم **1V** أم **1 kV**.

يجب أن نميز بين هذه الحالات وبين حالات عشوائية بطبيعتها ولها خواص إحصائية. ولكن إذا عدنا لحل مشكلة الشك، فهناك عدة طرق نذكر منها:

التحديث البايسي Bayesian updating

في هذه الطريقة نفترض أن بإمكاننا إسناد قيمة احتمال لكل فرضية، وأنه يمكن تحديث هذه الاحتمالات في ضوء الأدلة التي نحصل عليها والتي تعزز أو تدحض الفرضيات. لقد جرى تطبيق هذه الطريقة على نظام خبير ناجح قديم هو **PROSPECTOR**، الذي كان يعنى بدراسة أطياف المعادن من معطيات جيولوجية.

المبدأ هو التالي: لنفترض لدينا فرضية **H** احتمالها المبـدئي **a priori probability** هو **p(H)**. ولنفترض لدينا الدليل **E**، ونريد حساب الاحتمال الشرطي $p(H \mid E)$ (وهو احتمال صحة الفرضية بوجود الدليل)، إذا كنا نعلم $p(E \mid H)$ (احتمال وجود الدليل بافتراض صحة الفرضية). وتقول القاعدة:

$$p(H \mid E) = \frac{p(H) \times p(E \mid H)}{p(E)}$$

ويمكن أن نكتب هذه العلاقة بطريقة أكثر تفصيلاً كما يلي:

$$p(H \mid E) = \frac{p(H) \times p(E \mid H)}{p(H) \times p(E \mid H) + p(\sim H) \times p(E \mid \sim H)}$$

حيث نأخذ بالاعتبار احتمال الفرضية **p(H)** واحتمال نفيها **p(~H)** حيث

$$p(\sim H) = 1 - p(H)$$

يمكن أن يعطي الخبير الاحتمالات الموجودة في مقام **P(H|E)** من دراسات إحصائية أو من خبرته.

إن من حسنات هذه الطريقة أنها تحقق ما يلي:

- تعتمد على نظرية إحصائية مبرهنة.
- تتطلب التقانة احتمالات استنتاجية وليس استقرائية. فالمستثمر يعطي احتمال الدليل (الأعراض)، باعتبار الفرضية (السبب) معلومة، وليس العكس.
- يمكن تحديث احتمال الفرضية حين يتوفّر أكثر من دليل.

أما من سيئاتها:

- ضرورة معرفة الاحتمالات الأولية.
- ضرورة حساب الاحتمالات الشرطية أو قياسها.
- قيمة احتمال حقيقة ما، لا يعطي أي معلومة عن دقتها.

نظرية اليقين certainty theory

وهي نسخة مطورة عن التحديث البايسي، استُخدمت في نواة النظام الخبير **EMYCIN (Essential MYCIN)** المعتمد على النظام الطبي **MYCIN**. هذا النظام يساعد في تشخيص الأمراض الالتهابية. في هذه الطريقة يمكن التعبير عن فرضيات مشكوكة كما يلي:

إذا كانت الفرضية **H** صحيحة فإن معامل اليقين لها **C(H)=1**.

إذا كانت الفرضية **H** خاطئة فإن معامل اليقين لها **C(H)=-1**.

إذا كانت الفرضية **H** غير معلومة فإن معامل اليقين لها **C(H)=0**.

ويمكن إسناد معامل يقين **certainty factor (CF)** إلى كل قاعدة، يعبر عن مدى ثقتنا بهذه القاعدة. ويمكن أن يتغير هذا المعامل ليعكس مستوى ثقتنا بالدليل فنكتب القيمة الجديدة لمعامل اليقين هذا كما يلي:

$$CF' = CF \times C(E)$$

فإذا كان الدليل موجوداً كان $C(E)=1$ وكان $CF'=CF$.

لتحديث يقين فرضية **H** بوجود دليل **E** $C(H|E)$، يمكننا تطبيق التوابع التالية:

$$\begin{aligned}
&if\ \ C(H)\geq 0\ and\ CF'\geq 0:\\
&\quad C(H|E)=C(H)+[CF'\times(1-C(H))]\\
&if\ \ C(H)\leq 0\ and\ CF'\leq 0:\\
&\quad C(H|E)=C(H)+[CF'\times(1+C(H))]\\
&if\ \ C(H)\ and\ CF'\ have\ opposite\ signs:\\
&\quad C(H|E)=\frac{C(H)+CF'}{1-\min(|C(H)|,|CF'|)}
\end{aligned}$$

نتحقق أن $C(H|E)$ يحقق الخواص التالية:

- مستمر
- قيمه بين **-1** و **+1**.
- إذا كان **C(H)=1** أو **CF'=1** فإن **C(H/E)=1**
- إذا كان **C(H)= -1** أو **CF'= -1** فإن **C(H/E)= -1**
- في حالات التناقض (أي **C(H)= -CF'**) فإن **C(H/E)=0**
- إذا كان **C(H)=0** فإن **C(H/E)=CF'**.

إذا وجد لدينا عدة أدلة E_1, E_2 ..., فإننا نكتب:

$$\begin{aligned}
&C(E_1\ AND\ E_2)=\min[C(E_1),C(E_2)]\\
&C(E_1 OR\ E_2)=\max[C(E_1),C(E_2)]\\
&C(\sim E)=-C(E)
\end{aligned}$$

نظرية الإمكان: المجموعات العائمة والمنطق العائم

وسنأتي على ذكرها بالتفصيل في الفصول القادمة.

الفصل الرابع

النظم الخبيرة المعتمدة على الأطر

1. الأغراض والأطر والبرمجة الغرضية التوجه

حين نصمم نظاماً، نسعى في غالب الأحيان إلى تقسيم مسألة معقدة إلى مسائل أبسط. إن مفهوم الأغراض ومفهوم الأطر هما طريقتان مرتبطتان لتحقيق مثل هذا الهدف، دون المساس بتكاملية النظام **integrity** عامة.

إن البرمجة المعتمدة على الأطر **(Frame-based programming)** مرتبطة ببناء نظام قواعد معرفة وتنظيمها. أما البرمجة الغرضية التوجه **Object Oriented Programming (OOP)** فهي تُستخدم في العديد من النظم البرمجية ومنها النظم الذكية. لقد جرى تطوير البرمجة الغرضية التوجه كطريقة أفضل للبرمجة في حين جرى تطوير الأطر كطريقة تعبيرية لتمثيل المعلومات وتنظيمها.

تسعى كلا التقنيتين إلى تصميم برمجيات قابلة للصيانة والتكيف وإعادة الاستخدام. وهي تقسم العالم إلى مكونات جزئية. في النظم المعتمدة على الأطر، يمكن تمثيل كل من مكونات العالم بإطار يتضمن معلومات عن هذا المكون. تُعد الأطر غير فعالة **passive** لكونها تشبه مداخل قواعد المعطيات ولا تنفذ أي مهام بذاتها. الأغراض مشابهة للأطر، وتختلف عنها في كونها قادرة على إنجاز بعض الأفعال والمهام. فهي حين تتلقى تعليمات يمكنها تنفيذ بعض الأفعال الواقعة ضمن تعريفها.

إضافة إلى فوائد البرمجة الغرضية التوجه المذكورة سابقاً، فإن هذه البرمجة طريقة طبيعية لتمثيل الكثير من مسائل العالم الواقعي حاسوبياً ونمذجتها. من اللغات التي تدعم البرمجة الغرضية التوجه **C++**، و **Smalltalk** و **Java** و **CLOS (Common Lisp Object System)**. هذه اللغات جميعها تتضمن: تجريد المعطيات والوراثة والكبسلة (إخفاء المعطيات) والربط الديناميكي.

2. تجريد المعطيات

يسمح تجريد المعطيات بتعريف أنواع متشابهة من المعطيات تسمى الصفوف **classes**، وهي تشكل نماذج للأغراض. فالأغراض التي تمثل الفكرة نفسها أو المفهوم نفسه أو الغرض الفيزيائي نفسه تُجمع معاً كصفوف. وفي هذا السياق النوع والصف مترادفان.

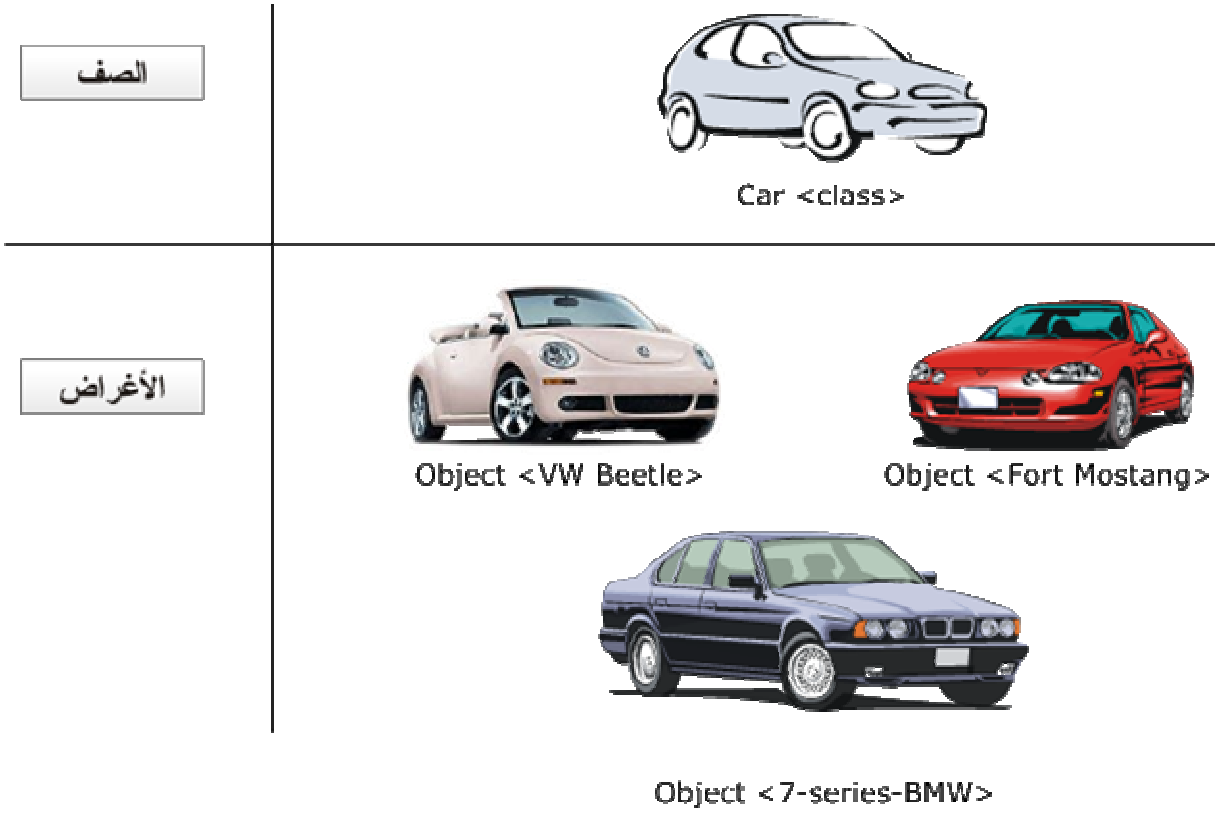
تتضمن معظم لغات البرمجة الحاسوبية أنواعاً مضمنة **built-in** من المعطيات كالأعداد

الصحيحة والأعداد الحقيقية، أما اللغات الغرضية التوجه فتتضمن أيضاً نوعاً إضافياً للمعطيات هو الصفوف، وهي تُعامل على نحو يشبه إلى حد كبير أنواع المعطيات المضمنة الأخرى. أحياناً، تسمى الصفوف التي يعرفها المستثمر أنماط معطيات مجردة **abstract data types**، وهي يمكن أن تكون صفوفاً لها دور خاص في مجال معين.

يتضمن تعريف الصف: اسم الصف وواصفاته **attributes** وعملياته وعلاقاته بالصفوف الأخرى.

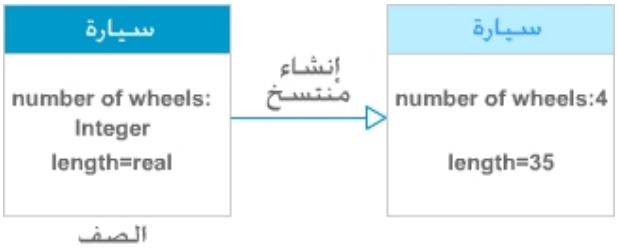
من الأمثلة على الصفوف: صف الأعداد الطبيعية الكبيرة، صف الأشكال الهندسية السطحية.

بعد إنشاء الصف يمكن إنشاء أغراض منتسخة **instances** من هذه الصفوف، ولهذه المنتسخات خواص الصفوف نفسها. على سبيل المثال: سيارتي هي منتسخ من الصف سيارة. إن هذا امتداد لمفهوم المتغير **variable** ونمط المعطيات في اللغات التقليدية. فبعد إنشاء الصفوف، يمكننا استخدام خواص هذه الصفوف بإنشاء منتسخات منها.



واصفات الصفوف هي مقادير كمية تصف منتسخات الصف. وأحياناً نعرفها على أنها شقوب **slots** نسند إليها قيماً. نصرح فقط عن اسم الصف وأنواع الواصفات حين نصرح عن الصف، وأحياناً نعطي قيماً للواصفات. يتصرف الصف كنموذج **template**، وحين ننشئ منه منتسخاً

فإن المنتسخ يتضمن أسماء الواصفات، أما قيم هذه الواصفات فيمكن أن تُعطى أو تُحسب لكل منتسخ، كما يمكن أن تؤخذ من الصف كقيم مغتفلة **by default**. يمكن أن تكون الواصفات من أي نمط وقد تكون أنماطاً مجردة (صفوف).



لننظر في المثال التالي:

CLASS: Boarding Pass
Carrier: [Str]
Name: [Str]
Flight: [Str]
Date: [Str]
Seat: [Str]
From: [Str]
To: [Str]
Boarding: [Number]
Gate: [Number]

الشكل **-1-** صف بطاقة صعود الطائرة **Boarding Pass**

هذا تمثيل صف بطاقة الصعود إلى الطائرة التي تتضمن الواصفات التالية على الترتيب: الشركة الناقلة، اسم المسافر، رقم الرحلة، تاريخ الرحلة، رقم المقعد، مدينة الانطلاق ومدينة الوصول (وجميعها من النوع **string**)، وساعة الصعود إلى الطائرة ورقم بوابة الانطلاق (وهما من النوع **integer**). ولكل مسافر بطاقة صعود إلى الطائرة هي منتسخ من هذا الصف، وأحد هذه المنتسخات هو التالي:

INSTANCE: Air New Zealand Boarding Pass
Carrier: Air New Zealand
Name: Mr. White
Flight: NZ 101
Date: 13 Dec
Seat: 25A
From: Melbourne
To: Christchurch
Boarding: 1800
Gate: 7

الشكل **-2-** منتسخ من صف بطاقة الصعود إلى الطائرة

وهذا المنتسخ هو لبطاقة صعود إلى الطائرة من شركة نيوزيلاندا. قيم واصفاتها هي التالية:

- الشركة الناقلة: طيران نيوزيلاندا
- اسم المسافر: السيد "وايت".
- رقم الرحلة: **NZ 101**.
- تاريخ الرحلة: **12/23**.
- رقم المقعد: **25A**.
- مدينة الانطلاق: ميلبورن.
- مدينة الوصول: كريست تشيرش.
- ساعة الصعود إلى الطائرة: السادسة مساء.
- بوابة الانطلاق: البوابة رقم **7**.

في لغة البرمجة **C++**، تسمى الواصفات أعضاء معطيات **data members**، ونفترض أن جميع أعضاء المعطيات هي واصفات منتسخات، إلا إذا صرحنا عنها على أنها سكونية **static**، فعندها تُخزن في موضع الذاكرة نفسه لجميع منتسخات الصف، وتوجد منها نسخة واحدة مهما كان عدد المنتسخات. ولذلك فإن أعضاء المعطيات السكونية هي واصفات للصفوف، وأعضاء المعطيات الأخرى واصفات للمنتسخات.

العمليات أو الطرائق أو التوابع الأعضاء، هي مجموعة العمليات أو التوابع التي يستطيع الغرض تنفيذها. في المثال التالي: الطريقة باللون الوردي، وهي عبارة عن حساب قيمة تخفيض سعر البطاقة، هذه الطريقة تُنفذ حين يكون عمر المسافر أقل من **25** سنة، فتُخفض قيمة البطاقة إلى النصف، وإلا فلا يوجد أي تخفيض.

```
 (Airline-Ticket
<Passenger Person>
<Discount
[when-needed
{if Passenger:Age < 25 then Passenger:Discount=50% else
        Passenger:Discount=0%
        Discount=Passenger:Discount}]>)
<Price number>
)
(Person
<Name String>
<Age number>
<Discount [when-needed
{if Age < 25 then Discount=50% else Discount=0%}]>
)
(John
```

```
<INSTANCE-OF Person>
<Name “John”>
<Age 24>
<Discount [when needed
….)
(Ticket1010
<INSTANCE-OF Airline-Ticket>
<Discount [when-needed…..>
<Price 1000>
)
Rule:
IF Airline-Ticket:Discount=50% THEN  Airline-Ticket:Passenger:Discount=50%
…
IF Ticket1010:Discount=50% THEN    Ticket1010:John:Discount=50%
```

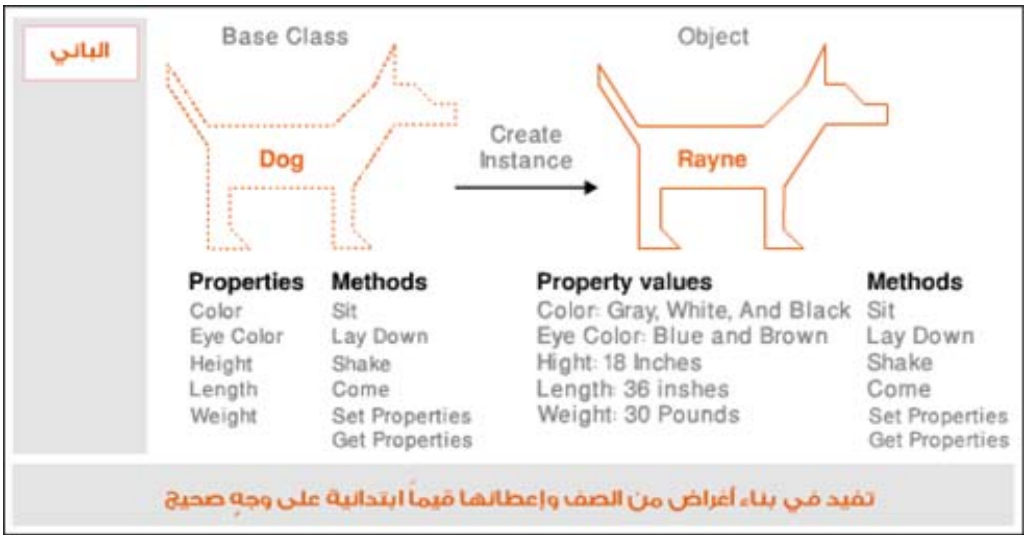
الشكل -3- مثال غرض يبين واصفات وطرائق (الطرائق باللون الأزرق)

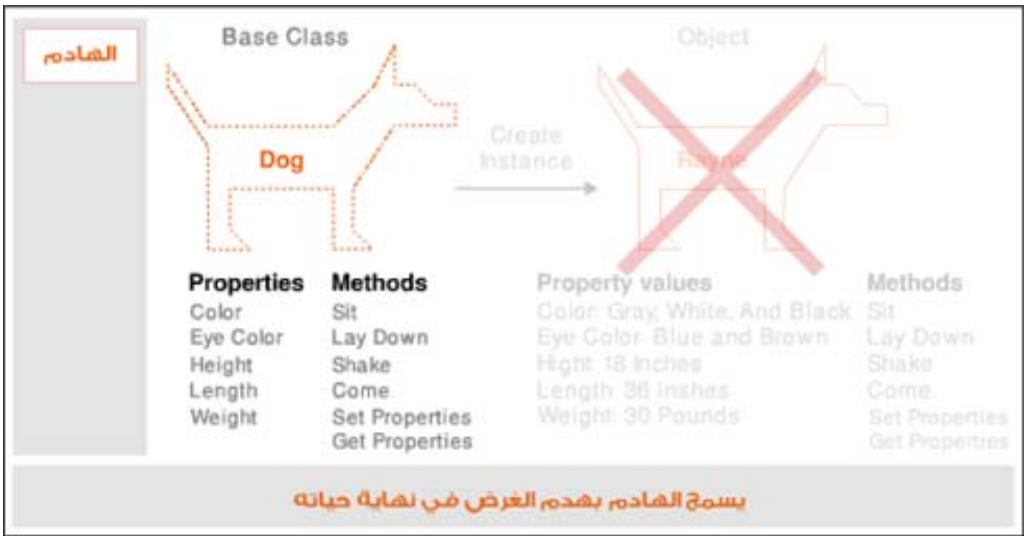
أما حول كيفية إنشاء منتسخات جديدة أو حذفها، فسنأخذ مثالاً ما يجري في **C++**، لإنشاء منتسخ جديد يمكن أن نكتب التعليمات التالية:

```
Myclass*  myinstance;
        // declare  a pointer  to objects of  type Myclass
myinstance  =  new  Myclass;
        // pointer  new points to a new instance
```

حين نحذف منتسخاً فإن تحرير الذاكرة التي كان يستخدمها من مسؤولية المبرمج. في لغات البرمجة الغرضية التوجه الأخرى تجري عملية تحرير الذاكرة تلقائياً في ما يسمى تجميع القمامة **garbage collection**. في لغة **C++**، إذا أردنا حذف المنتسخ **myinstance** نكتب صراحة:

```
Detete myinstance
```





وثمة طرائق يمكن أن تُنفذ لدى إنشاء المنتسخ تسمى البانيات **constructors** وأخرى لدى حذفه تسمى الهادمات **destructors**. الرماز التالي يبين مثالاً على ذلك:

```
// class definition:
class Sonic_pulse
{
        protected:
                float  amplitude;
        public:
                Sonic_pulse(float initial_amplitude);          // constructor
                ~Sonic_pulse();                                // destructor
};
// constructor definition:
Sonic_pulse::Sonic_pulse(float initial_amplitude)
{
        amplitude=initial_amplitude;                           // set up initial value
}
// destructor definition:
```

```
Sonic_pulse::~Sonic_pulse()
{
                // perform any tidying up that may be necessary before
                // deleting the object
}
```

الشكل -4- مثال برمجي يبين استخدام الباني **constructor** والهادم **destructor**

يستدعى الباني **constructor** فور إنشاء منتسخ جديد، كما يستدعى الهادم **destructor** فور حذفه.

على سبيل المثال يمكن إنشاء منتسخ جديد ضمن الرماز بإحدى الطريقتين:

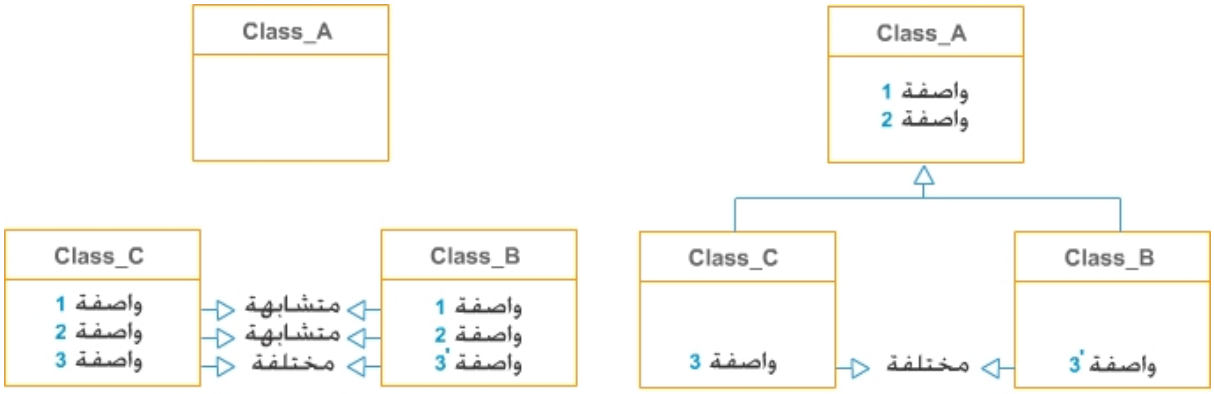
```
Sonic_pulse* myinstance = new Sonic_pulse(131.4);     //technique 1

or:

Sonic_pulse myinstance(131.4);                        //technique 2
```

في الطريقة الأولى يجب حذف المنتسخ صراحة من قبل المبرمج أما في الطريقة الثانية فإنه يحذَف آلياً حين يصبح خارج مدى التحكم في البرنامج.

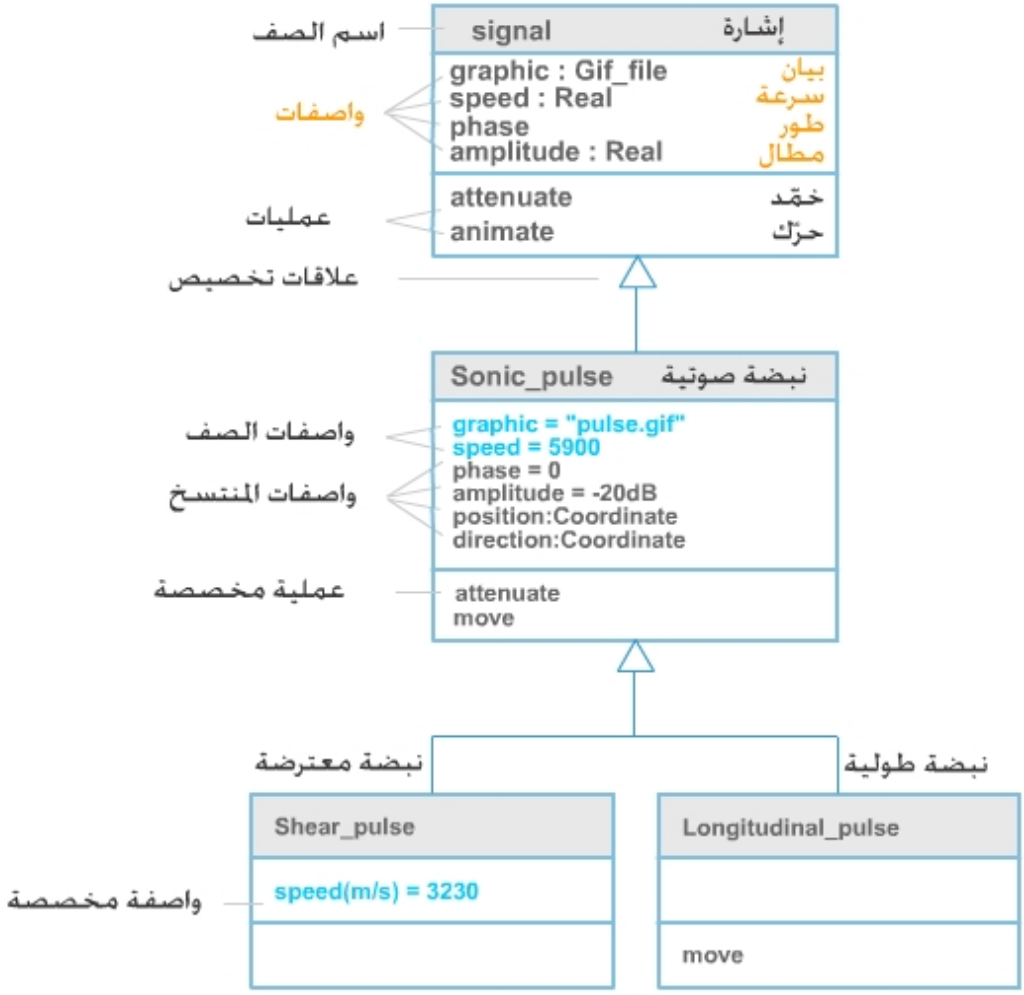
3. الوراثة

في كثير من الحالات، يكون لدينا مجموعة صفوف تتشارك في الكثير من المتشابهات. إن وضع هذه المتشابهات في كل صف يتطلب الكثير من التكرار. نتجنب هذه المشكلة باستخدام الوراثة **inheritance** أو الاشتقاق **derivation**. نعرف صفاً رئيسياً يتضمن جميع الواصفات المتشابهة المشتركة، ثم نعرف صفوفاً جزئية **sub-classes** ترث جميع الواصفات المتشابهة، ونضيف لكل منها الواصفات الإضافية الخاصة بها. الصف الرئيسي يسمى الصف العلوي من الصف الجزئي **super-class** أو تعميم الصفوف الجزئية. العلاقة بين الصفوف الجزئية والعلوية هي علاقة "من النوع" أو **"a-kind-of"**. العبارات التالية تبين هذه العلاقة:

- **a subclass is-a-kind-of superclass;**
- **an offspring is-a-kind-of parent;**
- **a derived class is-a-kind-of base class;**
- **a specialized class is-a-kind-of generalized class;**
- **a specialized class inherits from a generalized class.**



حين نعرف صفاً مشتقاً، يكفي ذكر الصف الأساسي الذي يرث منه الواصفات والطرائق، إضافة إلى فروقه عن ذلك الصف. في لغة **C++**، لا يرث الصف المشتق لا الباني ولا الهادم ولا المعامل **overloaded**. جميع الواصفات والطرائق التي تُذكر في الصف المشتق تسمى مخصصة، ونسمي الصف المشتق صفاً مخصصاً.يبين الشكل التالي مثالاً على الوراثة:



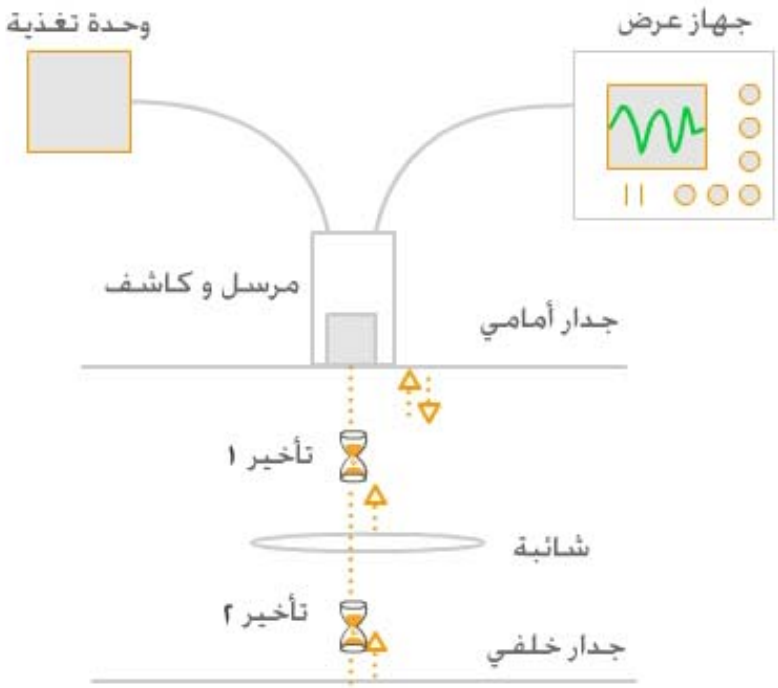
الشكل **-5-** مثال يبين الوراثة في الصفوف

يبين هذا المثال تعريف النبضة الصوتية **sonic_pulse**، وصفّها العلوي إشارة **signal**، وصفّيها الجزئيين: نبضة طولية **longitudinal_pulse**، ونبضة معترضة **shear_pulse**. الواصفات: بيان، وسرعة، وطور ومطال وهي جميعها معرفة في الصف الأساسي وترثها جميع الصفوف المشتقة. المتغير بيان **graphic** يأخذ قيمة مغتفلة عند مستوى النبضة الصوتية. المتغير سرعة **speed** يأخذ قيمة جديدة في حالة النبضة المعترضة تختلف عن قيمته التي يرثها من صفه العلوي.

إن وراثة العمليات مشابهة لوراثة الواصفات. العملية **move** معرفة على مستوى النبضة الصوتية، ويعاد تعريفها على مستوى النبضة الطولية.

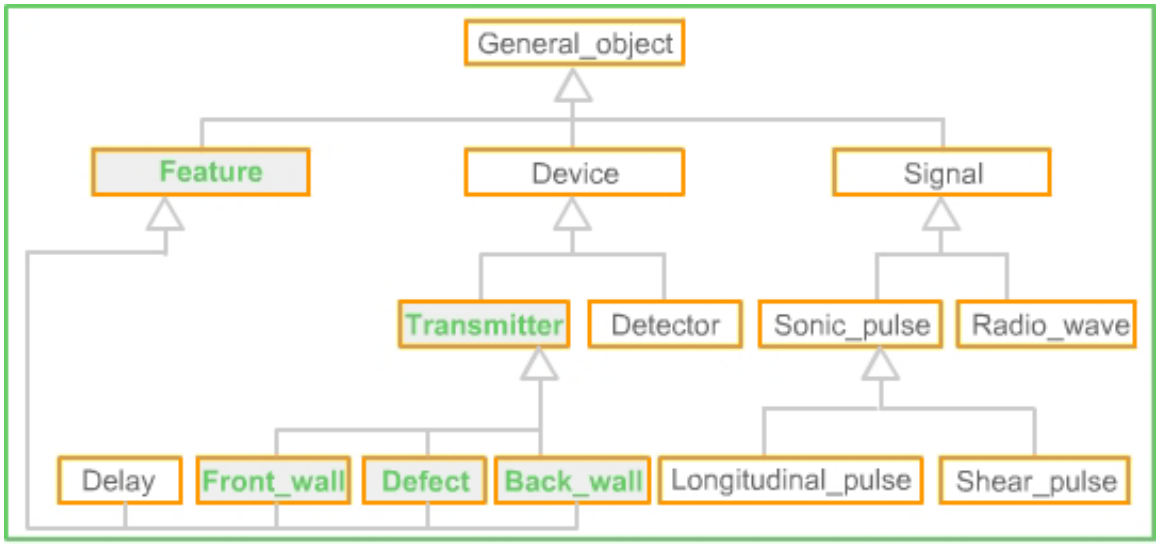
يبين الشكل **-6-** المثال نفسه ولكن بنظرة أشمل تبين صفوف محاكاة جهاز التصوير بالأمواج فوق الصوتية. الجهاز نفسه مبين بالشكل **-7-**.

الشكل**-6-**: علاقة تراتبية بين الصفوف في حالة الوراثة المفردة



الشكل -7- جهاز كشف الشوائب بالأمواج فوق الصوتية، التأخيران يدلان على عمق الشائبة وعمق الجدار الخلفي

تسمح بعض اللغات الغرضية التوجه بتعريف وراثة متعددة **multiple inheritance** حيث للصف أكثر من صف أب، وعوضاً عن أن تكون العلاقة بين الصفوف تراتبية هرمية تصبح شبكة **network** كما في المثال التالي المعدل عن الوراثة المفردة.

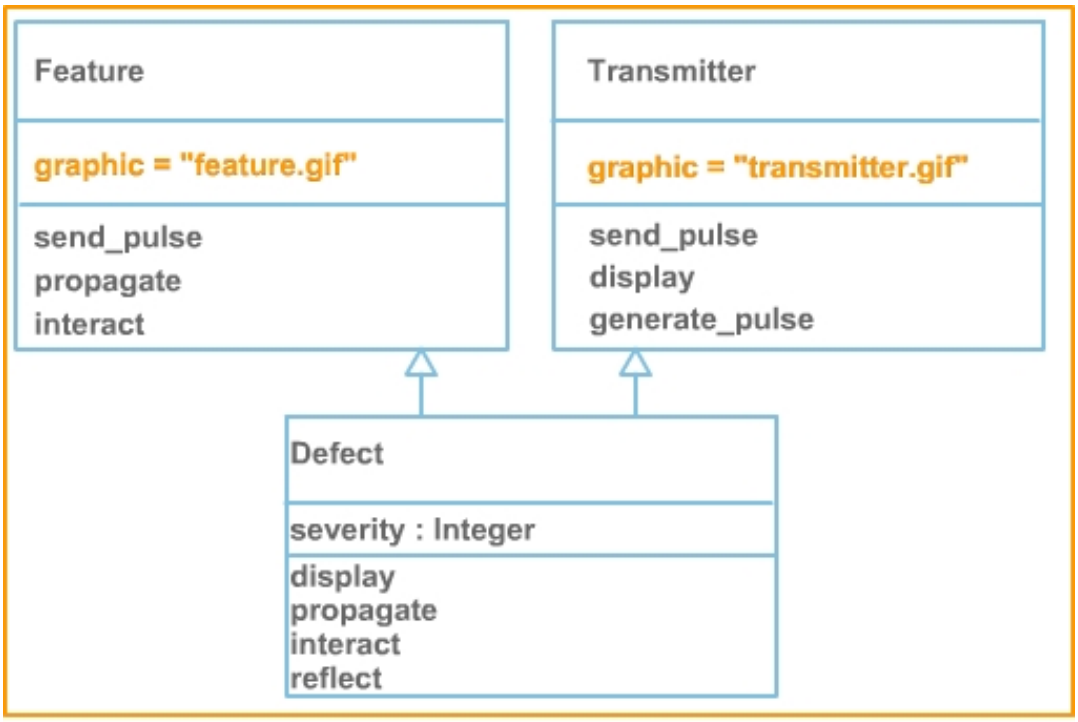


الشكل -8- علاقة بين الصفوف في حالة وراثة متعددة

في هذا التمثيل، الموجة الصوتية يمكن أن ترتد عن الجدار الأمامي **front_wall** أو عن الجدار الخلفي **back_wall** أو عن الشائبة **defect**. لذلك، فهذه الصفوف بحاجة إلى توليد أمواج، لذلك

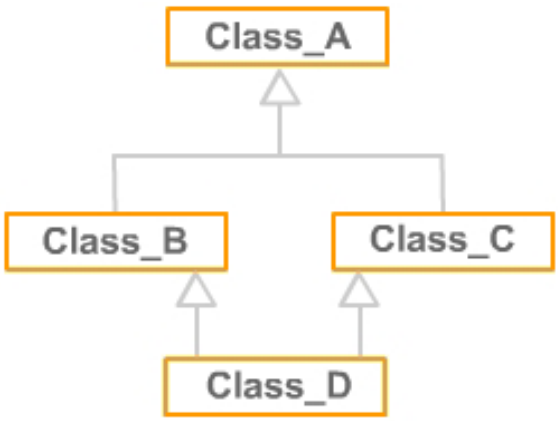
ترث هذه الإمكانية عن المرسل **transmitter**، إضافة إلى وراثتها من صف السمات **features**.

في حالة الوراثة المتعددة، قد يرث الصف واصفة لها الاسم نفسه من أبويه ولكن لها قيمة مختلفة من كل أب، هنا نحتاج لحل التعارض، إما باتباع استراتيجية أخذ الواصفة من الأب الأقرب أو الأب الأبعد أو ندع المبرمج يقرر صراحة أي القيم يريد اعتمادها للواصفات. كما في مثال وراثة **graphic** في الشكل **-9-**:



الشكل **-9-** تضارب بسبب الوراثة المتعددة

يمكن للصف أيضاً أن يرث من صف آخر عبر أكثر من طريق. يسمى هذا الوراثة المتكررة **repeated inheritance**. انظر المثال التالي:



الشكل **-10-** الوراثة المتكررة

تسمح لغة **C++**، في حالة المثال السابق، إما بإعطاء نسختين من الصف **A** إلى الصف **D**، نسخة عن **B** ونسخة عن **C**. أو بنسخة واحدة إذا جرى التصريح عن الصفين **B** و **C** عن أن صفّهما الأساسي الافتراضي هو الصف **A**.

4. الكبسلة

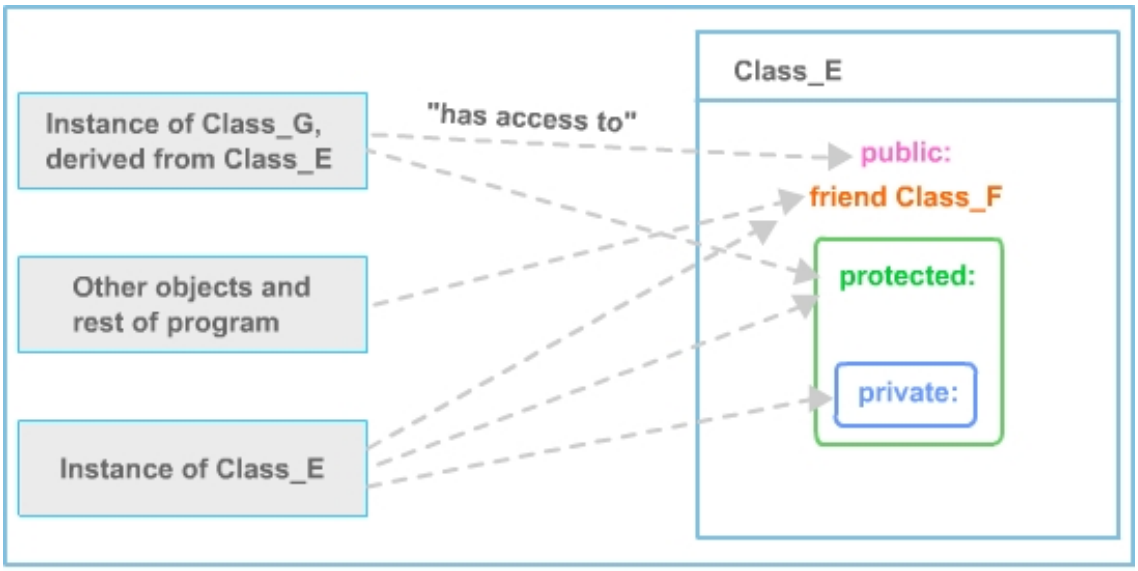
الكبسلة **encapsulation** أو إخفاء المعلومات، هي مصطلح يستخدم للتعبير عن أن واصفات المنتسخ وطرائقه، التي تعرف غرضاً، تنتمي لهذا الغرض ولا تنتمي لأي غرض غيره. نقول عن الواصفات والطرائق في هذه الحالة أنها خاصة **private** وأنها مكبسلة داخل الغرض.
واصفات الصف لا يمكن أن تكون مكبسلة ضمن أي منتسخ، لأنها مشتركة بين جميع منتسخات الصف الواحد. تسمح لغة **C++** بتحصيص مستوى نفاذ لكل التوابع الأعضاء والمعطيات، يأخذ إحدى القيم الأربعة التالية:

خاص **private**: (قيمة مغتفلة **default**)، يمكن النفاذ إليها من توابع الصف نفسه فقط.

محمي **protected**: يمكن النفاذ إليها من توابع الصف نفسه، والصفوف المشتقة منه فقط.

عمومي **public**: يمكن النفاذ إليها من أي جزء من البرنامج.

صديق **friend**: يمكن النفاذ إليها من توابع صفوف مسماة فقط.



الشكل **-11-** التحكم في النفاذ بلغة **C++**

5. لغة النمذجة الموحدة UML والربط الديناميكي

تزود لغة النمذجة الموحدة **UML** بوسيلة لتحديد العلاقات بين الصفوف والمنتسخات وتمثيلها بيانياً. لقد مر معنا سابقاً علاقتين هما:

- التخصيص/ التعميم **specialization/ generalization.**
- الاستنساخ **instantiation.**

تسمى منتسخات الصف أو الصف الجزئي زبائن **clients** لأنها ترث من هذه الصفوف. ثمة علاقات جديدة ثلاثة هي:

- التجميع **aggragation**

وهي حين يتكون الغرض من مجموعة أغراض جزئية. الملكية فيها ضعيفة فالغرض الجزئي يمكن أن ينتمي لأكثر من غرض أساسي. مثال: مجتزآت المكتبات البرمجية.

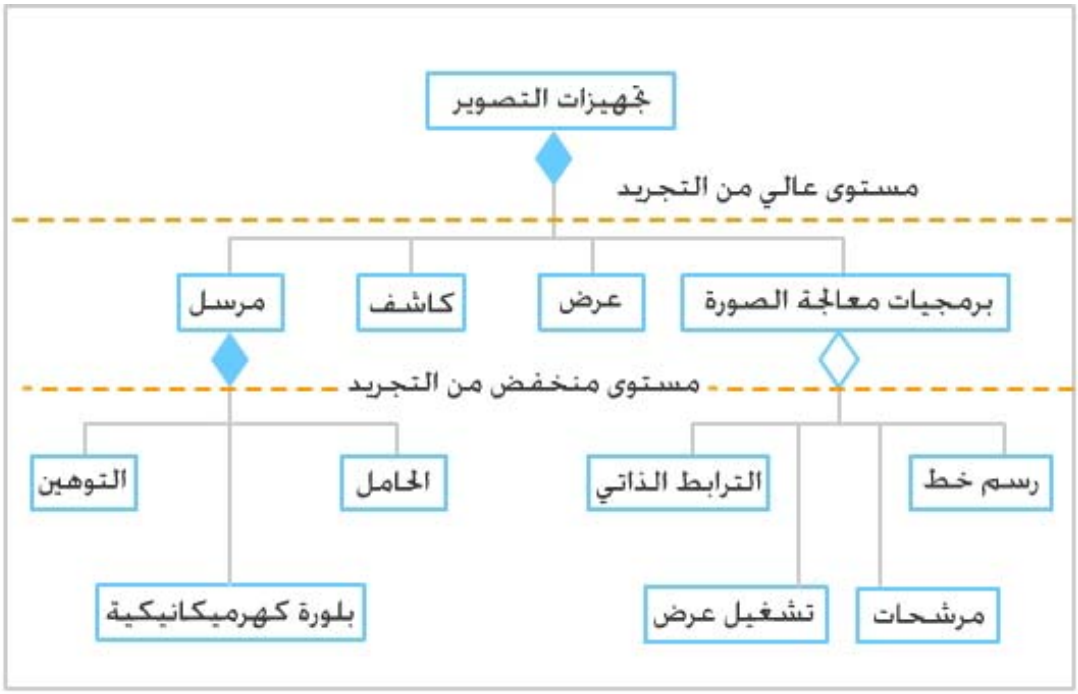
- التركيب **composition.**

حالة خاصة من التجميع، علاقة الملكية فيها قوية. مثال: تتركب السيارة من محرك وهيكل وعجلات وأبواب ومقاعد و...منتسخ السيارة يضم منتسخات عن جميع المكونات.

- الإرفاق **association.**

علاقات ضعيفة **loose** بين الأغراض. مثال: علاقة بين السيد **X** والكلب الذي يمتلكه. مثال آخر: أغراض يمكنها أن ترسل رسائل إلى أغراض أخرى. مرسل الرسائل يسمى ممثل فعال **actor**، ومستقبلها يسمى مخدم **server**، أما من يرسل الرسائل ويستقبلها معاً فيسمى وكيل **agent.**

تتضمن لغة **UML** رسماً بيانياً للعلاقات بين الصفوف. يشار إلى علاقة التركيب بخط فيه معين أسود، وللتجميع بخط فيه معين مفرغ أبيض، أما الإرفاق فيشار إليه بخط عادي بلا معين. انظر المثال التالي:



الشكل **-12-** علاقات التركيب (المعينات السوداء المملوءة) والتجميع (المعينات البيضاء المفرغة) على مخطط **UML**

أما الربط الديناميكي **dynamic binding** أو الربط المتأخر **late binding**، وهو المكون الرابع الضروري للغات الغرضية التوجه، فيشير إلى أنه على الرغم من أن الصفوف الآباء تكون معروفة لحظة ترجمة البرنامج **compilation**، إلا أن الصفوف الفعلية المشتقة لا ترتبط بأغراض لحظة الترجمة وإنما يؤجل ربطها إلى وقت التنفيذ، وهذا مايسمى الربط الديناميكي.

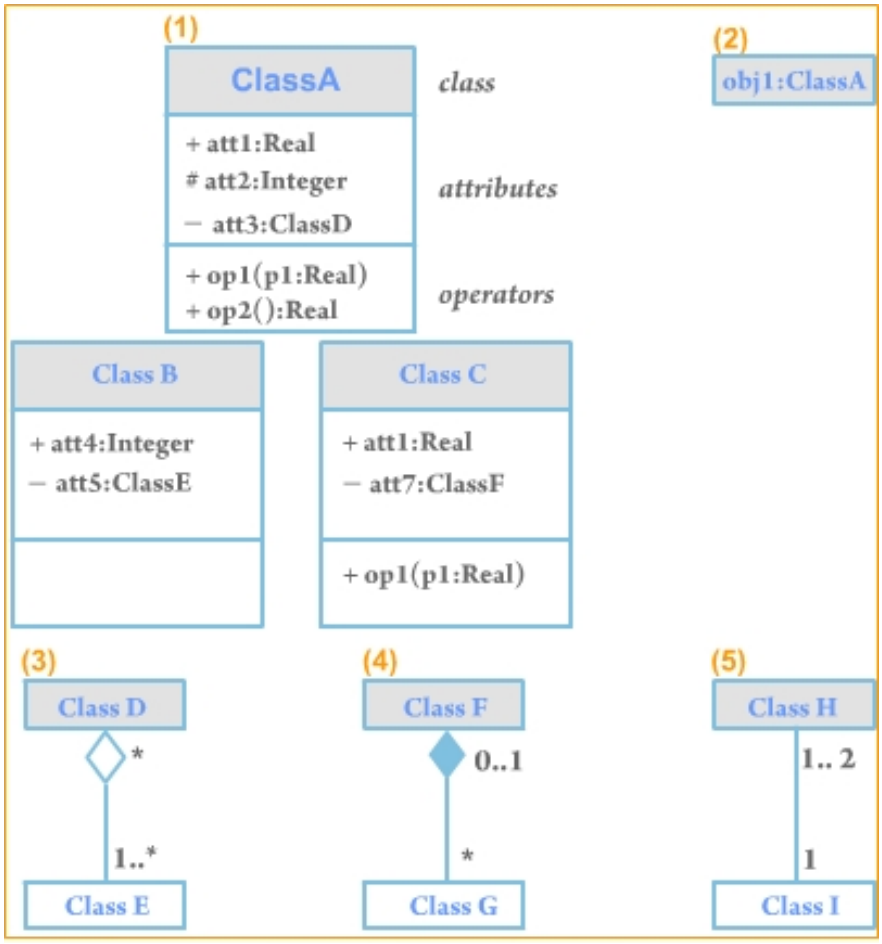
لننظر في الشكل التوضيحي التالي الذي يبين علاقات بين الصفوف بلغة **UML**. في هذا الشكل:

(a)- الصف **A** صف معمم للصفين **B** و **C**.

(b)- **obj1**: منتسخ من الصف **A**. **(c)**- تجميع **(d)**-تركيب **(e)**- إرفاق

الأرقام تشير إلى عدد مسموح من الأغراض: * أي عدد

التحكم في النفاذ: **+**: عمومي، #: محمي، -: خصوصي.



الشكل -13- علاقات الصفوف بلغة **UML**

في هذا المثال، قد تكون لدينا طريقة **operation** لحساب قيمة ما، من واصفات الغرض **att**. حين التنفيذ تؤخذ الواصفة عدداً طبيعياً **integer** أو حقيقياً **real**، بحسب الصف المشتق. وبذلك قد يكون للطريقة الاسم نفسه ولكن معنى مختلف بحسب الصف المشتق.

6. مواضيع أخرى

العفاريت demons

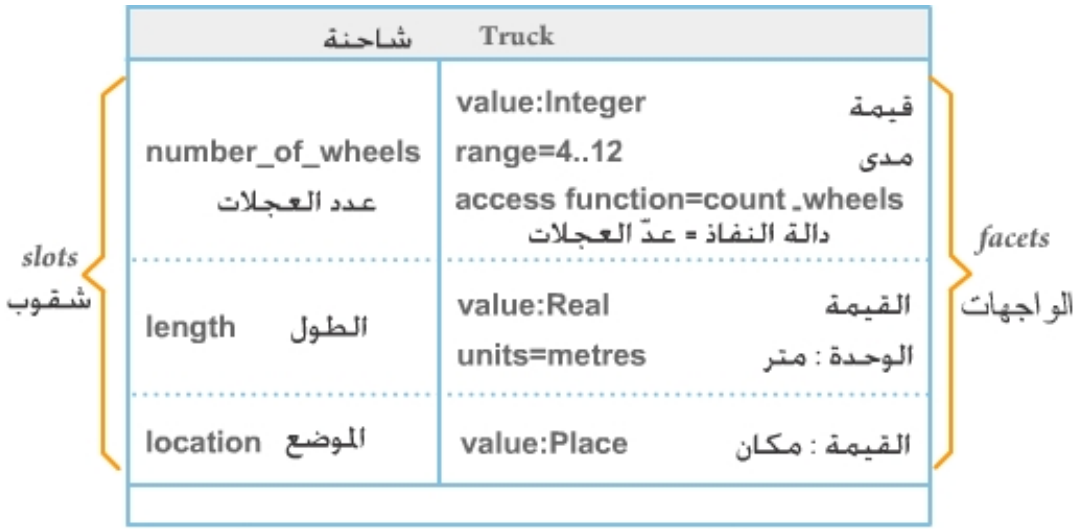
حتى الآن، افترضنا أن الطرائق تُستدعى صراحة في البرامج، وهي يمكن أن تغير في المعطيات. ثمة مفهوم جديد هو مفهوم القيمة الفعالة **active value**، والعفاريت **daemons or demons**، وهي تسمح لنا بسلوك معاكس للسابق: استدعاء التوابع هو أثر جانبي ناتج عن النفاذ إلى معطيات واصفات تسمى القيم الفعالة. مثال: استخدام **demon** في بناء مقياس ضمن واجهة مستثمر بيانية. يمكن إنشاء غرض المقياس لمراقبة واصفة في غرض فيزيائي مثل الفلطية في محول. حالما تتغير قيمة الفلطية، يجب على المقياس أن يحدث الرسم، بقطع النظر عن السبب

الذي أدى إلى تغيير القيمة. انظر الشكل التالي:

الشكل-14-: استخدام العفريت **demon** لمراقبة قيمة فعالة

النظم المعتمدة على الأطر

ترتبط النظم المعتمدة على الأطر **frame-based** بالنظم الغرضية التوجه **object-oriented**. الأولى نشأت وتطورت نتيجة بحوث في الذكاء الصنعي، في حين نشأت الثانية في هندسة البرمجيات. تسمح الأطر بتمثيل المعرفة وتنظيمها. وهي تستخدم سمات مفتاحية غرضية التوجه لتمثيل المعلومات، ولكنها لا تمتلك سلوكات الأغراض وطرائقها. بما أنه يمكن إنشاء علاقات بين الأطر، فيمكن استخدامها لنمذجة الروابط الفيزيائية والمجردة. وبنفس الطريقة التي نشتق بها منتسخات الأغراض من صفوف الأغراض، يمكننا اشتقاق منتسخات الأطر من صفوف هذه الأطر، مستفيدين من الوراثة. لقد عرف بعضهم النظم المعتمدة على الأطر باعتبارها نظم توفّر تجريد المعطيات والوراثة دون أن تتطرق إلى الكبسلة أو الربط الديناميكي. ذلك أن ليس للأطر مالك للتوابع فيها، ولهذا فإنها بنى غير فعالة **passive**. ومع ذلك تبقى الأطر أداة فعالة لتمثيل نظام قواعد معرفة. لننظر في المثال التالي:



الشكل -15- مثال: تمثيل معرفة معتمد على الأطر

في المثال السابق نعرف الإطار شاحنة. ويمكننا أن نأخذ منتسخاً عنه هو "شاحنتي"، نملأ الشقوب بالقيم المناسبة من هذا الإطار المنتسخ. يمكن أن يكون الشقب: عدداً، أو صفة، أو مجال عددي، أو إجرائية أو إطار آخر، أو أي شيء يسمح به التنجيز البرمجي.

تسمح بعض النظم المعتمدة على الأطر، بوضع قيم متعددة في الشقوب، المعلومات المختلفة التي نرفقها بشقب معين تسمى واجهاته. يمكن أن نرفق بكل واجهة قيمة معينة. لنأخذ شقب عدد العجلات في المثال السابق: يمكن أن يكون له قيمة مغتفلة **(4)**، أو أن نحدد مدى عدد العجلات، أو أن نستدعي تابعاً لعد العجلات.

وفيما يلي مثال يبين الوراثة بين الأطر:

```
/* define a frame */
frame vehicle;
        default location is garage and
        default number_of_wheels is 4
        and default mass_in_tonnes is 1 .

/* define another frame */
frame truck is a kind of vehicle;
        default mass_in_tonnes is 10 .

/* create an instance of frame truck */
instance my_truck is a truck .

/* create another instance of frame truck */
instance your_truck is a truck;
number_of_wheels is my_truck`s number_of_wheels + 2 .
```

الشكل **-16-** الوراثة بين الأطر

يمكن أن نطبق الأطر على أغراض مادية كالشاحنة، أو مجردة كالخطة والتصميم.

3. السيناريو scripts

نوع خاص من الأطر، هو تسلسل من الأطر يصف أفعالاً أو سلاسل أفعال. بما يشبه فيلماً سينمائياً أو شريط فيديو.

المثال التالي يبين مشهد الدخول إلى المطعم

Generic RESTAURANT Frame
Specialization-of: Business-Establishment
Types:
range: (Cafeteria, Fast-Food, Seat-Yourself, Wait-To-Be-Seated)
default: Seat-Yourself
if-needed: IF plastic-orange-counter THEN Fast-Food,
IF stack-of-trays THEN Cafeteria,
IF wait-for-waitress-sign or reservations-made THEN Wait-To-Be-Seated,
OTHERWISE Seat-Yourself.
Location:
range: an ADDRESS
if-needed: (Look at the MENU)
Name:
if-needed: (Look at the MENU)
Food-Style:
range: (Burgers, Chinese, American, Seafood, French)
default: American
if-added: (Update Alternatives of Restaurant)
Times-of-Operation:
range: a Time-of-Day
default: open evenings except Mondays
Payment-Form:
range: (Cash, CreditCard, Check, Washing-Dishes-Script)
Event-Sequence:
default: Eat-at-Restaurant Script
Alternatives:
range: all restaurants with same Foodstyle
if-needed: (Find all Restaurants with the same Foodstyle)

الشكل **-17-** سيناريو الدخول إلى مطعم

الفصل الخامس

المنطق العائم

المنطق العائم

يشهد المنطق العائم والمجموعات العائمة استخداماً متزايداً في الكثير من المجالات العلمية، سيما تطبيقات الذكاء الصنعي حيث بدأنا نشهد نظماً عائمة. هذا الفصل هو مقدمة للتعريف بمفردات المنطق العائم والمجموعات العائمة، وكيفية التعامل معها، وإجراء الحسابات باستخدامها، وذلك تمهيداً للفصل القادم الذي يتحدث عن النظم الخبيرة العائمة.

1. المنطق التقليدي والمنطق العائم

نبدأ بالمنطق التقليدي ثم ننتقل منه إلى المنطق العائم الذي يشكل تعميماً لمفهوم المنطق التقليدي. يعرف المنطق بأنه "العلم الذي يتحرى المبادئ التي تحكم الاستدلال الصحيح والموثوق". تُعتَبر الفرضيات العنصر الأساسي في المنطق، والفرضية هي عبارة تؤكد أو تنفي شيئاً ما، وبذلك يمكن توصيفها بأنها صحيحة أو خاطئة. على سبيل المثال، نذكر العبارات التالية:

- اسم الأستاذ أحمد
- عمر الأستاذ ثلاثون عاماً
- اسم الأستاذ أحمد، وعمره ثلاثون عاماً.
- ...

لكل عبارة في المنطق التقليدي القيمة "صح" أو "خطأ"، ولا يمكن أن يكون لها أي قيمة بينهما. نسند عادة قيماً رقمية للحقائق السابقة، فنسند القيمة **1** لـ "صح" والقيمة **0** لـ "خطأ".

من المبادئ الأساسية للمنطق التقليدي، قانون استبعاد الوسط **Excluded Middle**، الذي ينص على أن كل عبارة هي "صح" أو "خطأ" ولاحالة ثالثة لها، وقانون عدم التناقض **Non-Contradiction** الذي ينص على أنه لا يمكن للعبارة أن تكون "صح" و"خطأ" بآن معاً. سنشير إلى الفرضيات بأحرف كبيرة: **P, Q, R,...**، ونشير إلى قيم الحقيقة لها بالأحرف نفسها أي **P, Q, R,...**.

ثمة نوعان لقيم الحقيقة، الإمكان **possibility**، وهو يقيس إلى أي حد يحتمل **might** أن تكون الفرضية صحيحة، والحاجة **necessity**، وهو يقيس إلى أي حد يجب **must** أن تكون الفرضية صحيحة. في غياب أي معطيات خارجية إمكان الفرضية **1** وحاجتها **0**. و لأننا نتعامل مع مسائل من الحياة الواقعية، فما يهمنا هو الحاجة أي الوصول إلى نتائج يجب أن تكون صحيحة لأنها مدعومة بالمعطيات، وليس نتائج يحتَمل أن تكون صحيحة.

يمكننا الحصول على فرضيات معقدة بدءاً من فرضيات بسيطة، وذلك باستخدام الروابط: **AND, OR, NOT, IMPLY** (العطف "و"، الفصل "أو"، النفي، والاقتضاء).

الشكل التالي يبين جدول الحقيقة للفرضيات المركبة في المنطق التقليدي:

P	Q	¬P	P∧Q	P∨P	P⇒Q	P⇔Q
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

الشكل **-1-** جدول الحقيقة لبعض الروابط في المنطق التقليدي.

أما في المنطق العائم، فترتبط قيمة الفرضيات المركبة بالفرضيات البسيطة كما يلي:

معامل النفي NOT:

$$NOT(P) = 1-P$$

معامل العطف AND وله عدة أشكال نسميها على الترتيب معامل زاده **Zadeh**، المعامل الاحتمالي بفرض استقلالية الفرضيتين، ومعامل الفرق المحدود:

Zadeh operator: (P AND Q)= min (P, Q)
Probabilistic operator: (P AND Q)=P*Q (if independent)
Bounded difference operator: (P AND Q)=max(0, P+Q-1)

معامل الفصل OR وله عدة أشكال نسميها على الترتيب معامل زاده، المعامل الاحتمالي بفرض استقلالية الفرضيتين، ومعامل المجموع المحدود:

Zadeh operator: (P OR Q)= max (P, Q)
Probabilistic operator: (P OR Q)=P+Q-P*Q (if independent)
Bounded sum operator: (P OR Q)=min(1, P+Q)

كما نرى، لكل صيغة **P AND Q** صيغة موافقة **P OR Q**، على الترتيب، نسميها المعامل المثنوي **dual operator**. ويمكننا اشتقاق كل صيغة موافقة من الصيغة الأصلية باستخدام علاقات دومورغان:

$$P \vee Q = \neg((\neg P) \wedge (\neg Q))$$
$$P \wedge Q = \neg((\neg P) \vee (\neg Q))$$

ولكن أي المعاملات نختار للعمليات **AND** و **OR**؟ لقد استخدم المعامل زاده لفترة طويلة وأثبت

نجاحه، فضلاً عن أن هذا المعامل هو نظرياً الخيار الأصح. ومع ذلك، يبقى اختيار المعاملات الأخرى ممكناً.

تمرين:

تأكد أنه في حال اخترنا المنطق التقليدي، أي كانت قيم **P** و **Q** صفراً أو واحداً، فإن جميع العلاقات باستخدام جميع المعاملات تبقى صحيحة.

معامل الاقتضاء imply ($\rightarrow$) أو ($\Rightarrow$) ويمكن حسابه بعدة طرق بالمنطق العائم، أشهرها:

$$(P \rightarrow Q) = 1 \text{ if } P \leq Q, \text{ else } (P \rightarrow Q) = 0$$

$$(P \rightarrow Q) = \min(1, 1 - P + Q)$$

$$(P \rightarrow Q) = \max(1 - P, \min(P, Q))$$

يمكن استنتاج علاقة الاقتضاء باستخدام المعاملات السابقة كما يلي:

$$P \rightarrow Q = (\text{NOT } P) \text{ OR } Q$$

نلاحظ أن معامل الاقتضاء يعد إمكاناً وليس حاجةً.

نعلم أن مبدأ مودس بوننس **Modus Ponens** يشير إلى أن:

$$P \text{ AND } (P \rightarrow Q)) \rightarrow Q($$

أي إذا كان لدينا **P** صحيحة والقاعدة $P \rightarrow Q$ فإننا نستنتج صحة **Q**، وهذه علاقة الاستنتاج.

في الحقيقة، إذا كانت معاملاتنا عائمة، فإن النتيجة محتملة وليست صحيحة بالضرورة، لذلك لايمكن استخدام مودس بوننس العائمة في النظم المعتمدة على القواعد.

يهتم المنطق العائم أيضاً بصحة الفرضيات، ولكن في العالم الواقعي، الفرضيات صحيحة جزئياً. إضافة إلى ذلك، فإننا نستخدم في غالب الأحيان، مفردات غير معرفة بدقة، فمثلاً حين نقول العبارة "جون رجل مسن" لانستطيع أن نجزم بصحتها إذا كان عمر جون **50** عاماً. لذلك، نفضل إعطاء قيمة الحقيقة لهذه العبارة، قيمةً ما في المجال **[0,1]**. وبذلك تكون قيمة الحقيقة لفرضية ما، هي قياس في المجال **[0,1]** لمدى تأكدنا من صحة الفرضية. ليست الفرضيات وحدها من

تملك قيماً لصحتها، وإنما للمعطيات أيضاً قيم حقيقة مرتبطة بها، وهي أيضاً ضمن المجال [0, 1].

الفرضيات في المنطق العائم صحيحة جزئياً، ذلك أن المعطيات التي نختبرها صحيحةٌ جزئياً، والمقارنات التي نجريها صحيحة جزئياً أيضاً.

2. المجموعات العائمة

لتكن لدينا المجموعة **X** المكونة من مجموعة من الأغراض نسميها المجموعة الكلية **universal set**، ونأخذ مجموعة المجموعات الجزئية منها. لتكن **A** مجموعة جزئية من **X**. نعرف لكل عنصر **x** من **X**، تابع الانتماء الذي يأخذ القيمة **1** إذا كان **x** ينتمي إلى **A** (نكتب **A(x)=1**)، والقيمة **0** إذا كان **x** لا ينتمي إلى **A** (نكتب **A(x)=0**). نسمي هذا التابع بالتابع المميز **characteristic function**. المجموعات العائمة توسع مفهوم التابع المميز ليأخذ أي قيمة في المجال **[0, 1]**.

نعرف المجموعة الجزئية العائمة **F** من **X** بتابع العضوية (الذي هو تعميم للتابع المميز)، الذي يأخذ قيمه في المجال **[0, 1]**، نسمي القيمة **F(x)** درجة عضوية العنصر **x** للمجموعة **F**، ونرمز لها غالباً **μ(x)**.

$$\mu_F(x)=1 \Rightarrow x \ \ belongs \ \ to \ \ F$$
$$\mu_F(x)=0 \Rightarrow x \ \ doesn't \ \ belong \ \ to \ \ F$$
$$0<\mu_F(x)<1 \Rightarrow x \ \ in \ \ F \ \ with \ \ membership \ \ \mu_F(x)$$

المجموعات المنفصلة العائمة

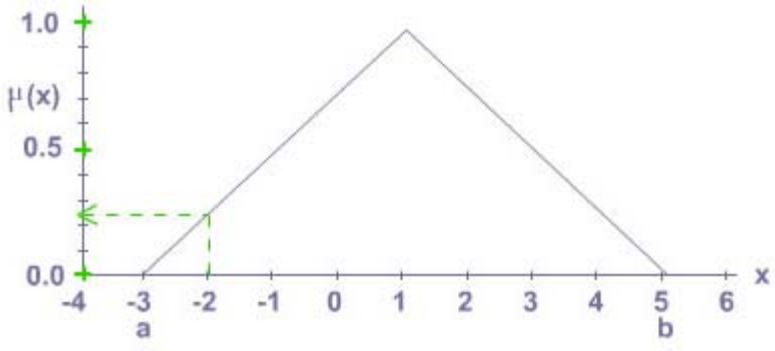
إذا كانت **X** مجموعة منتهية، وكانت **D** مجموعة جزئية منها، يمكننا كتابة مايلي:

$$D=\left\{\frac{\mu_1}{x_1},\frac{\mu_2}{x_2},...,\frac{\mu_n}{x_n}\right\}$$

حيث μ_1 هي قيمة عضوية x_1 في **D**، ونسميها أيضاً درجة العضوية. يبقى التعريف صحيحاً وإن لم تكن **X** منتهية ولكن **D** مجموعة جزئية منتهية منها. يمكن أن تكون المجموعات المنفصلة العائمة رقمية أو غير رقمية، بحسب الكميات التي يوصفها أعضاء هذه المجموعة.

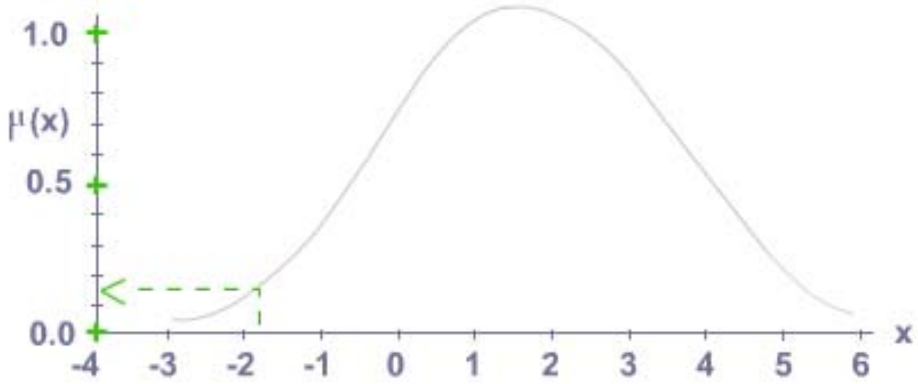
الأعداد العائمة

تمثل الأعداد العائمة أعداداً لسنا متأكدين من قيمها. إنها حالة خاصة من المجموعات العائمة، أعضاؤها أعداد من المستقيم الحقيقي. يمكن أن تكون الأعداد العائمة من أي شكل، ولكن عادة نتطلب أن تكون محدبة ولها مساحة منتهية. يمكن أن يكون شكل تابع العضوية مثلثاً أو على شكل جرس أو شبه منحرف. انظر الأشكال التالية للعدد العائم **1**:

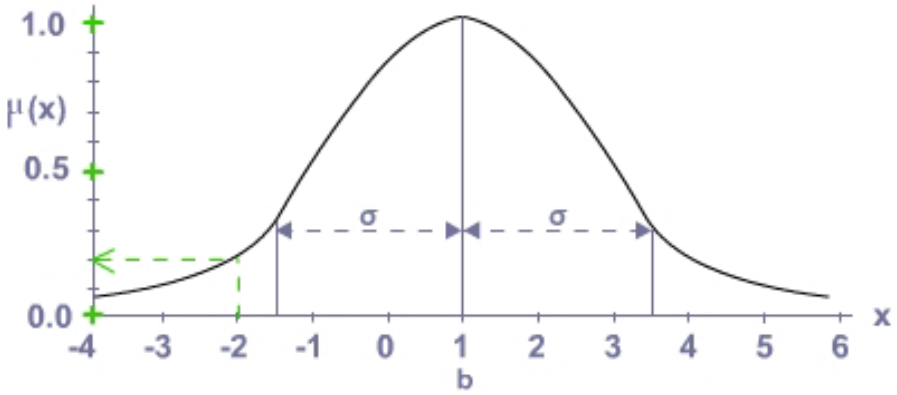


الشكل -2-: تابع العضوية المثلث للعدد **1**، عضوية **2-** تساوي **0.25** ونعرف تابع العضوية له كما يلي:

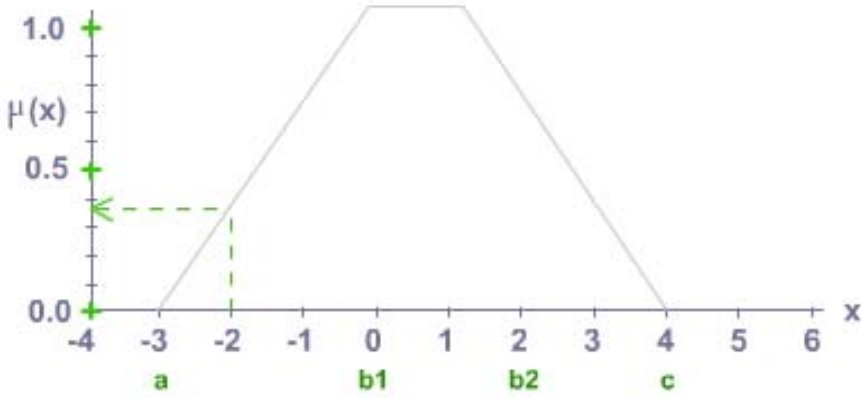
$$\mu(x)=\begin{cases}0, & x\le a\\ (x-a)/(b-a), & a<x\le b\\ (c-x)/(c-b), & b<x\le c\\ 0, & x>c\end{cases}$$



الشكل -3- تابع العضوية للعدد **1** على شكل حرف **S**، عضوية **2-** تساوي **0.1**



الشكل-4-: تابع العضوية للعدد **1** على شكل توزع طبيعي، عضوية **2-** تساوي **0.2**.



الشكل-5-: تابع العضوية للعدد **1** على شكل شبه منحرف، عضوية **2-** تساوي **0.34**.

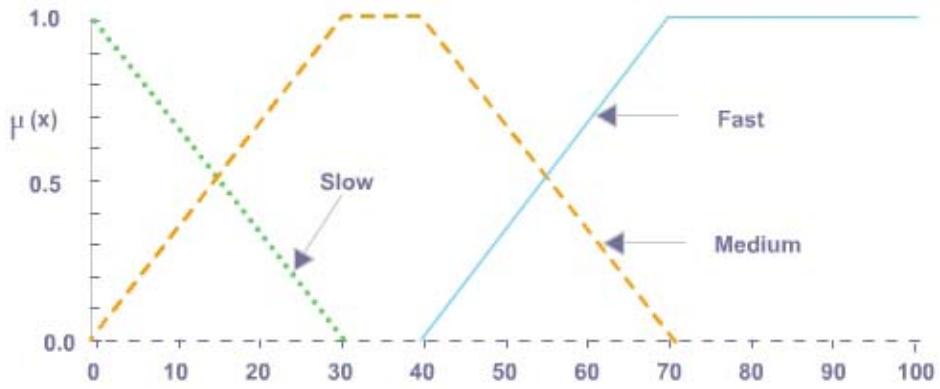
منطلق العدد العائم هو المجال بين النقطة التي تتزايد منها العضوية من الصفر، والنقطة التي تتناقص عندها العضوية لتعود إلى الصفر.

حين يقتصر منطلق العدد العائم على نقطة وحيدة نعود إلى تعريف العدد غير العائم، ونقول إن لدينا نقطة مفردة **singleton**.

المتغيرات اللغوية وتوابع العضوية

جرت صياغة المجموعات المنفصلة العائمة كمتغيرات لغوية **linguistic variables**، تتكون من أسماء المجموعات المنفصلة العائمة، أسماء الأعضاء هي المتغيرات اللغوية، ولكل قيمة لغوية، تابع عضوية خاص بها.

لنأخذ، على سبيل المثال، المتغير اللغوي "السرعة"، أعضاؤه هي: "بطيء"، و "متوسط"، و "سريع". يبين الشكل التالي توابع العضوية لهذه الأعضاء، لسرعة سيارة مقدرة بالواحدة كم/سا:



الشكل -6- توابع العضوية للمتغير اللغوي "سرعة".

3. العلاقات العائمة

أحياناً، نتعامل مع صفيفة **array** (أو مصفوفة **matrix**) من قيم الحقيقة، كما في حالة تقطيع تابع العضوية، نسميها المصفوفات العائمة. إن جمع المصفوفات العادية مماثل لعملية **OR** العائمة على المصفوفات العائمة، كذلك الضرب (بثابت) في المصفوفات العادية مماثل لعملية **AND** العائمة على المصفوفات العائمة.

عملية تركيب المصفوفات **composition** مماثلة لضرب المصفوفات، وتخضع للقيود نفسها من حيث عدد عناصر المصفوفات لنتمكن من تركيبها. لتكن لدينا المصفوفتان العائمتان **A** (بسطر واحد و **m** عمود)، و **B** (بـ **m** سطر و **n** عمود). نرمز لناتج تركيب المصفوفتين بـ **A°B**، وهي مصفوفة بسطر وحيد و **n** عمود، بحيث يكون:

$$C_{ij} = (a_{i1} \wedge b_{1j}) \vee (a_{i2} \wedge b_{2j}) \vee ... \vee (a_{in} \wedge b_{nj})$$

إذا أخذنا المنطق العائم **min-max**، بحيث **OR** هي تابع القيمة العظمى **max** و **AND** هي تابع القيمة الصغرى **min**.

لنأخذ المثال التوضيحي التالي:

$$A = \begin{bmatrix} 0.2 & 0.4 & 0.6 \\ 0.3 & 0.6 & 0.9 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 & 1 \\ 0.7 & 0.5 \\ 1 & 0 \end{bmatrix}, \quad C = A \circ B = \begin{bmatrix} 0.6 & 0.4 \\ 0.9 & 0.5 \end{bmatrix}$$

من جهة أخرى، إذا أخذنا مجموعتين X و Y فإننا نقصد بالمجموعة $X \times Y$ هي مجموعة الأزواج المرتبة (x, y)، x في X و y في Y. العلاقة العائمة R على $X \times Y$ هي مجموعة جزئية عائمة من $X \times Y$

بحيث يكون لدينا لكل زوج (x, y) عدد يتراوح بين 0 و 1، يقيس العلاقة بين x و y.

يمكن أن تُعطى العلاقة العائمة على شكل مصفوفة حين تكون المجموعات منفصلة discrete، أو على شكل علاقة تحليلية للمجموعات المستمرة. لتوضيح الفكرة نأخذ المجموعة X={John, Jim, Bill}، والمجموعة Y={Fred, Mike, Sam}، وهما مجموعتان كل منهما بأسماء ثلاثة أشخاص، ولنأخذ العلاقة بين X و Y هي علاقة "التشابه بين الأشخاص"، فإن العلاقة بينهما تعطى بالمصفوفة التالية:

	Fred	Mike	Sam
John	0.2	0.8	0.5
Jim	0.9	0.3	0.0
Bill	0.6	0.4	0.7

الشكل -7- مثال على علاقة التشابه العائمة بين أشخاص

4. القرضيات العائمة fuzzy propositions

الشكل العام لقرضية عائمة بسيطة هو معامل مقارنة بين A و B: **(A (comparison operator) B**، حيث A و B معطيات متوافقة يمكن مقارنتها.

لنأخذ على سبيل المثال، العددين العائمين x وله القيمة 3، بقيمة حقيقة 0.8، وy الذي له القيمة 3 بقيمة حقيقية 0.7 ولنأخذ القرضية العائمة: **A: (x=y)** ولنبحث عن قيمة الحقيقة لهذه القرضية. بما أن قيمة الحقيقة للمقارنة تساوي 1 فإن قيمة الحقيقة للقرضية A تساوي **min(0.8, 1, 0.7)= 0.7**.

ونستطيع أن نكتب ما يلي:

Tv(x (comparison operator) y)= min (tv(x), tv(comparison), tv(y))

حيث tv هي قيمة الحقيقة **truth value**. إن قيمة الحقيقة لمعاملات المقارنة المعروفة $<, \leq, =, \geq, >, \neq$ هي إما 0 أو 1 دائماً.

يمكن اختبار أعضاء المجموعات العائمة المنفصلة بالطريقة نفسها. فإذا كانت لدينا القرضية

B =(size is small) أي الحجم **size** صغير **small**.

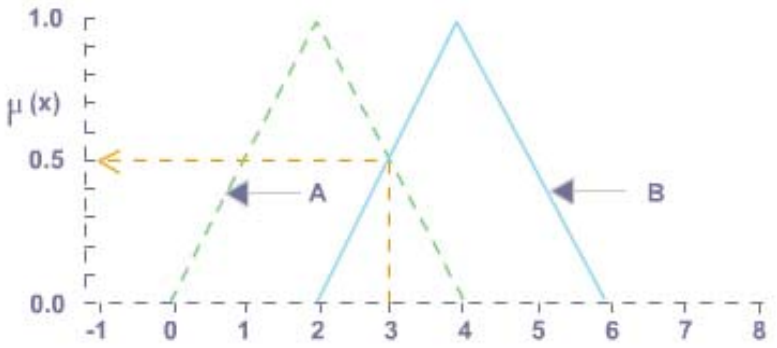
Size مجموعة عائمة منفصلة أحد أعضائها **small**. **Tv(small)** هي درجة عضوية **small** ضمن **size**، ولتكن **0.75** مثلاً. **Tv(size)=1**، وكذا قيمة الحقيقة لفعل الكون **is** هي **tv(is)=1**، وبذلك يكون لدينا:

Tv(B)=min (1, 1, tv(small))=0.75

فإذا أخذنا العلاقة **A≅B** أي **A** تساوي تقريباً **B** فإن قيمة الحقيقة لهذه الفرضية هي:

Tv(A≅B) =max (min (A(x), B(x))

فإذا كانت **A** و **B** كما في الشكل التالي:



الشكل -8- توابع العضوية للعددين العائمين **A** و **B**.

يتقاطع **A** و **B** في منطقة، تراوح قيمة العضوية فيها بين **0** و **0.5** وبذلك تكون القيمة العظمى له هي **0.5** وبالتالي فإن قيمة الحقيقة لـ **A≅B** هي **0.5**.

يمكن استخدام علاقات رياضية أخرى من أجل العلاقات: أصغر تقريباً >~ وغيرها.

5. التعويم وفك التعويم (إزالة التعويم)

التعويم

للتعويم **fuzzification** معنيان: إيجاد نسخة عائمة من المفهوم التقليدي غير العائم **crisp**، أو إيجاد درجة العضوية لقيمة لغوية لمتغير لغوي يوافق دخلاً رقمياً أو سلمياً أو عائماً. والمعنى الثاني هو المعنى المطلوب.

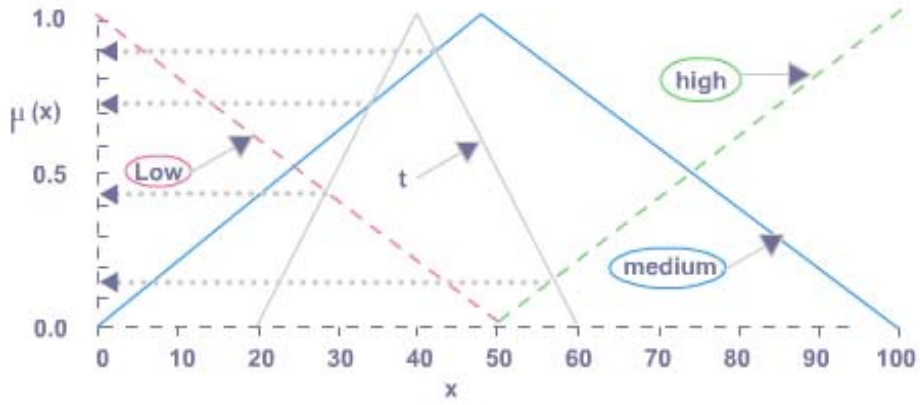
لنفترض رقماً افتراضياً **t** يأخذ قيماً بين **0** مئوي و **100** مئوي، ونريد إيجاد درجة انتماء قيمة لغوية لمتغير لغوي (وليكن درجة الحرارة) ويمكن أن يكون اسم المجموعة درجة الحرارة،

وأعضاؤها {منخفضة، وسط، مرتفعة} (**Temp ={low, medium, high}**)، وتابع العضوية لكل منها معرف على المجال **[0, 100]**.

إن عملية التعويم بسيطة، نفترض أن درجة العضوية هي قيمة الحقيقة للفرضية العائمة الموافقة

$$\mu(low) = tv\,(t \cong low)$$
$$\mu(medium) = tv\,(t \cong medium)$$
$$\mu(high) = tv\,(t \cong high)$$

حيث **μ(low)** هي درجة عضوية **low** في المتغير اللغوي الدال على درجة الحرارة **Temp**، والرمز ≅ يشير إلى المقارنة التقريبية بين المعاملات، كما عرفناه سابقاً. يبين الشكل التالي توابع العضوية لدرجة الحرارة **Temp** مع عدد عشوائي **t** جرى تعويمه في **Temp**.



الشكل -9- توابع العضوية للمتغير اللغوي درجة حرارة، وتعويم قيمة درجة الحرارة **t**.

توابع العضوية للمتغير اللغوي "درجة الحرارة"، الدخل الذي نود تعويمه هو الرقم العائم **t**. درجة العضوية لمنخفض **low** هو **0.43**، ولمتوسط **medium** هو **0.89**، ولعالي **high** هو **0.15**. تمثل هذه القيم أعلى قيم تقاطع بين **t** وكل من **low** و **medium** و **high**. وبذلك، تُعرف درجة الحرارة بالمجموعة العائمة التالية:

$$Tempreature = \left\{ \frac{0.43}{low}, \frac{0.89}{medium}, \frac{0.15}{high} \right\}$$

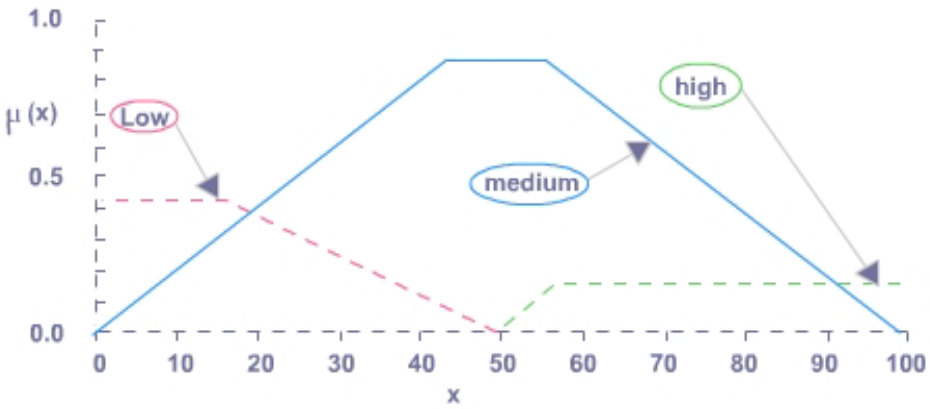
فك التعويم

بعد تنفيذ مجموعة قواعد في نظام خبير عائم، نتوصل إلى نتائج (نتائج القواعد) عائمة، نشير إليها بـ **C** وهي متغيرات لغوية، لقيمها درجات عضوية. ولكن، في غالب الأحيان، نحتاج إلى حساب قيم سلمية توافق قيم العضوية هذه. فك التعويم هو عملية حساب القيم السلمية من القيم العائمة. نحتاج إلى فك التعويم في التحكم العائم، لأن علينا إرسال النتيجة النهائية (بعد فك تعويمها) كإشارة إلى إجرائية التحكم.

إن فك التعويم أعقد بكثير من التعويم، وله عدة طرق، سنذكر الشائع منها. سنفترض أن درجات العضوية للمجموعات العائمة التي نود فك تعويمها معلومة.

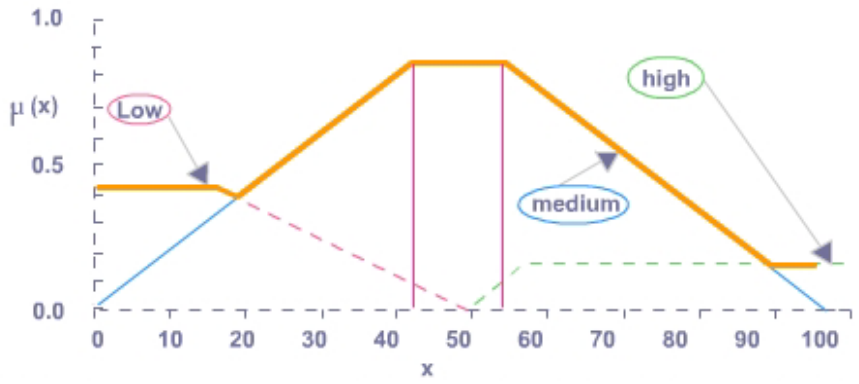
ليكن لدينا المتغير اللغوي **Lvariable**، أحد أعضائه القيمة **value** (مثال: القيمة قصير **value="short"** هي أحد أعضاء المتغير اللغوي طول **Lvariable = "length"**). نشير إلى عضوية عدد حقيقي **x** في القيمة **value** بـ **μ(x, value)**، ونشير إلى عضوية القيمة **value** في المتغير اللغوي **Lvariable** بـ **μ(value, Lvariable)**.

في الخطوة الأولى، نغير تابع العضوية لنعكس أن عضوية القيمة **value** في **Lvariable** ليست بالضرورة **1**. نحقق ذلك بعملية **AND** بين تابع العضوية **μ(x, value)**، وتابع العضوية **μ(value, Lvariable)**، أشهر استعمال لـ **AND** هو أخذ الـ **min** بين القيمتين، فنحصل على الشكل التالي لدرجة الحرارة، من الشكل السابق:



الشكل -10- توابع العضوية للمتغير اللغوي "درجة الحرارة"، معدلة بدرجة العضوية للقيم اللغوية، التابع **AND** هو **min**.

بعد ذلك، نوجد تابع عضوية وحيد لكامل المتغير اللغوي، التابع الأكثر استخداماً هو تابع **OR** بمفهوم الـ **max**، فنحصل على التابع الوحيد التالي:



الشكل -11- توابع العضوية المختارة للقيم اللغوية في المتغير اللغوي "درجة الحرارة"، **OR** هو **max**.

الخطوة الأخيرة هي إيجاد عدد وحيد متوافق مع تابع العضوية الأخير الذي حصلنا عليه، ليكون دخلاً إجرائية فك التعويم. ثمة عدة طرق لإيجاد مثل هذه القيمة منها: متوسط القيمة العظمى، متوسط القيمة العظمى الموزونة للمتوسط، وطريقة المراكز **centroid method** وهي الأكثر شهرة.

طريقة متوسط القيمة العظمى هي الأبسط، نرى مجال القيمة العظمى لتابع العضوية، نسمي x_{min} القيمة الصغرى لهذا المجال و x_{max} القيمة العظمى له. في المثال السابق **xmin=43** و **xmax=55**، والقيمة المتوسطة $\bar{X} = (x_{\max} + x_{\min})/2 = 49$ هي ناتج فك التعويم.

في طريقة متوسط القيمة العظمى الموزونة، نرى أن للتابع أكثر من قيمة عظمى، الأولى تمتد على المجال **[0, 15]** بدرجة عضوية **0.42**، والثانية تمتد على المجال **[43, 55]** بدرجة عضوية **0.88**، نأخذ قيمة فك التعويم هي التالية:

$$\bar{X} = (\frac{0+15}{2} * 0.42 + \frac{43+55}{2} * 0.88)/(0.42+0.88) = 35.6$$

طريقة المراكز، وهي المفضلة والأكثر استخداماً، تحسب ناتج فك التعويم كما يلي:

$$\bar{X} = \frac{\int_a^b x.\mu(x)}{\int_a^b \mu(x)} = 41.8$$

حيث نفترض المجال **[a, b]** هو مجال تعريف تابع العضوية الوحيد الممثل لجميع توابع العضوية للمتغير اللغوي (انظر الخطوة قبل الأخيرة).

6. جبر المجموعات العائمة

لتكن لدينا المجموعات العائمة A, B, C,... وهي مجموعات جزئية من X، ونريد أن نحسب A∪B و A∩B وغيرها. ما نستخدمه في المنطق العائم هو تعميم للمعاملات AND و OR، ويسميان على التتالي t-norm و t-conorm. فما هي هذه المعاملات؟

t-norm هو تابع T معرف على [0,1]×[0,1] ويأخذ قيمه في [0,1]. فإذا كان z=T(x,y)، فإن كلاً من x, y, z تنتمي إلى المجال [0,1]. ولجميع التوابع t-norms الخواص التالية:

1. T(x, 1) = x (boundary الحدود)
2. T(x, y) = T(y, x) (commutativity التبادلية)
3. if $y_1 \le y_2$, then $T(x, y_1) \le T(x, y_2)$ (monotonicity الاطراد)
4. $T(x, T(y, z)) = T(T(x, y), z)$ (associativity التجميعية)

إن t-norm هو تعميم لمفهوم AND حيث: tv(P AND Q)= T(tv(P), tv(Q)). لذلك، نجد ثلاثة t-norms أساسية بحسب تعريف الـ AND العائم، هي على التتالي:

$$T_m(x, y) = \min(x, y)$$
$$T_L(x, y) = \max(0, x + y - 1)$$
$$T_p(x, y) = x\ y$$
$$T^*(x, y) = \begin{cases} x & if \quad y = 1; \\ y & if \quad x = 1; \\ 0 & otherwise \end{cases}$$

ولدينا:

$$T^* \le T \le T_m \qquad \text{و} \qquad T^* \le T_L \le T_p \le T_m$$

فإذا كانت لدينا المجموعتان A, B فإن تابع العضوية للمجموعة C=A∩B هو التالي (بحسب دومورغان):

$$C(x) = NOT\ T(NOT\ A(x), NOT\ B(x)) = 1 - T(1 - A(x), 1 - B(x))$$

التابع t-conorm، بالطريقة نفسها، وهو يعمم مفهوم OR، ومعرف على [0,1]×[0,1] ويأخذ قيمه

في **[0,1]**. فإذا كان **C** هو **t-conorm** وكان **z=C(x,y)**، فإن:

1. **C(x, 1) = x (boundary الحدود)**
2. **C(x, y) = C(y, x) (commutativity التبادلية)**
3. **if** $y_1 \le y_2$ **, then** $C(x, y_1) \le C(x, y_2)$ **(monotonicity الاطراد)**
4. $C(x, C(y, z)) = C(C(x, y), z)$ **(associativity التجميعية)**

وأهم التوابع **t-conorms** هي:

$$C_m(x, y) = \max(x, y) \quad called \quad s\tan dard \quad union$$
$$C_L(x, y) = \min(1, x + y) \quad called \quad bounded \quad sum$$
$$C_p(x, y) = x + y - x\,y \quad called \quad a\lg ebraic \quad sum$$
$$C^*(x, y)\ called\ drastic\ union:$$
$$= \begin{cases} x & if \quad y = 0; \\ y & if \quad x = 0; \\ 1 & otherwise \end{cases}$$

ونعلم أن:

$$C_m \le C_p \le C_L \le C^* \qquad \text{و} \qquad C_m \le C \le C^*$$

فإذا كانت لدينا المجموعتان **A, B** فإن تابع العضوية للمجموعة **D=A∪B** هو التالي :

D(x) = C(A(x), B(x))

أما المجموعة المتممة للمجموعة العائمة **A** فنرمز لها **A^c**. ويكون لدينا:

$A^c(x)=1-A(x)$.

ويمكن بسهولة عبر الخواص التجميعية، تعميم تعريف **t-norms** و **t-conorms**، لأكثر من متحول. على سبيل المثال:

$$T_m(x_1, \ldots, x_n) = \min(x_1, \ldots, x_n)$$
$$C_m(x_1, \ldots, x_n) = \max(x_1, \ldots, x_n)$$

والجدير بالذكر أن **t-norms** و **t-conorms** تتمتع بخواص مثنوية بحيث:

$$C(x, y) = 1 - T(1 - x, 1 - y)$$

ملاحظة: قانون التوزيع بين الاجتماع والتقاطع ليس دائماً صحيحاً.

نشير إلى الاستدلال العائم بالمحاكمة التقريبية **approximate reasoning**. وهو يعرف كما يلي:

$$If \;\; X \;\; is \;\; A \;\; then \;\; Y \;\; is \;\; B$$
$$from \;\; X \;\; = A'$$
$$\inf er \;\; that \;\; Y \;\; = B'$$

يتضمن المرجع في الفصل الرابع منه مثالاً على حساب مثل هذا الاستدلال العائم.

7. الحساب العائم

عرفنا الأعداد العائمة على أنها مجموعات عائمة، ونود تعريف العمليات الحسابية عليها من جمع وطرح وضرب وتقسيم. يمكننا فعل ذلك بإحدى طريقتين:

- مبدأ التوسيع **extension principle**
- باستخدام حذف ألفا وحسابات المجال **alpha-cut** و **interval arithmetic**.

مبدأ التوسيع:

لنرمز بـ * إلى إحدى العمليات الحسابية الأربع: جمع، طرح، ضرب أو قسمة. ونريدحساب **P=M*N**، حيث **M** و **N** عددان عائمان. إذا كانت العملية هي القسمة، فعلينا أن نفترض أن ليس للصفر عضوية موجبة للمقسوم عليه **N**.

نعرف **P** كما يلي:

$$P(z) = \sup_{x,y} \{\min(M(x), N(x)) \mid x * y = z\}$$

لنفترض أن العملية هي الجمع، ونريد أن نحسب **P(z=4)**، علينا إيجاد جميع أزواج القيم **x** و **y** التي مجموعها **4**، ونحسب **M(x)** و **N(y)**، نأخذ القيم الصغرى لـ **M(x)** و **N(y)**، ثم نأخذ القيمة العظمى لهذه القيم الصغرى فتكون هي **P(4)**. هذه الطريقة صعبة التنفيذ حاسوبياً.

استخدام حذف ألفا وحسابات المجال:

- حذف ألفا

لتكن $\alpha \in (0,1]$ (مجال مفتوح عند الصفر مغلق عند الواحد)، نسمي حذف ألفا للمجموعة **A** (نكتبها $A[\alpha]$) المجموعة غير العائمة المعرفة كما يلي: $\{x \mid A(x) \geq \alpha\}$.

أي هي مجموعة العناصر x من X التي قيمة عضويتها تساوي على الأقل ألفا. أما قيمة **A[0]** فنعرفها على أنها قاعدة العدد العائم **(base** أو **support)**. في الشكل -2- حيث مثلنا العدد بمثلث فإن $A[1]=[a,b]$، وفي الشكل -5- حيث مثلنا العدد بشبه منحرف فإن $A[1]=[a,c]$. في حالة التمثيل بتوزيع طبيعي فإن القاعدة غير منتهية، ولكن إذا افترضنا أن **A(x)=0** حين تكون **x ≤ a- 3σ** وحين تكون **x ≥ a+3σ** فنستنتج أن $A[0]=[a-3\sigma,a+3\sigma]$.

إن نواة العدد العائم هي **A[1]** وقاعدته المجال **A[0]**. نقول عن المجموعة العائمة أنها طبيعية إذا كانت نواتها غير خالية. سنفترض أن جميع أعدادنا العائمة طبيعية. وبذلك يكون حذف ألفا للعدد العائم مجالاً محدوداً مغلقاً. ونكتب بشكل عام: $A[\alpha]=[a_1(\alpha),a_2(\alpha)]$ حيث a_1, a_2 هي حدود المجالات وهي تابعة لـ α. فإذا كانت $A[0]=[a_1,a_2]$ فإننا نكتب:

- **A > 0** إذا كان **a1 > 0**.
- **A ≥ 0** إذا كان **a1 ≥ 0**.
- **A < 0** إذا كان **a2 < 0**.
- **A ≤ 0** إذا كان **a2 ≤ 0**.

والآن إذا كان لدينا المجالان **[a, b]** و **[c, d]** فإننا نعمم مفهوم العمليات الحسابية * إلى ما يلي:

$$[a,b]*[c,d]=\{x*y \mid x\ in\ [a,b],\ y\ in\ [c,d]\}$$

وبذلك يكون:

$$[a,b]+[c,d]=[a+c,\ b+d]$$
$$[a,b]-[c,d]=[a-c,\ b-d]$$
$$[a,b]\div[c,d]=[a,\ b].\left[\frac{1}{d},\frac{1}{c}\right]$$

إذا لم يكن الصفر ضمن المجال **[c, d]**.

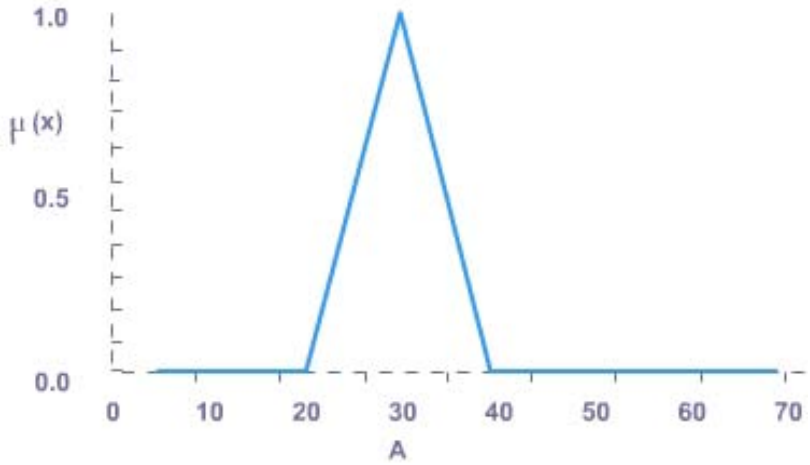
$$[a,b].[c,d]=[\kappa,\nu],\ for$$
$$\kappa=\min\{ac,ad,bc,bd\}$$
$$\nu=\max\{ac,ad,bc,bd\}$$

في الحالة الخاصة حين **a > 0** و **c > 0** يكون **[a, b].[c, d] = [ac, bd]**.

أما لمقارنة الأعداد العائمة، فإننا نعرف التساوي كما يلي:

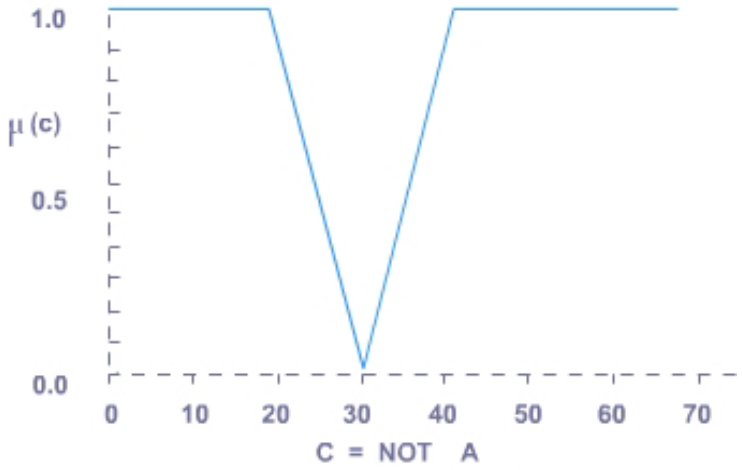
$$tv(A = B) = \sup_{x,y}\{\min(M(x), N(y)) | x = y\}$$

فإذا أردنا أن نقارن **A~< B** (**A** أصغر تقريباً من **B**)، **A** هو العدد العائم المبين بالشكل -12-.

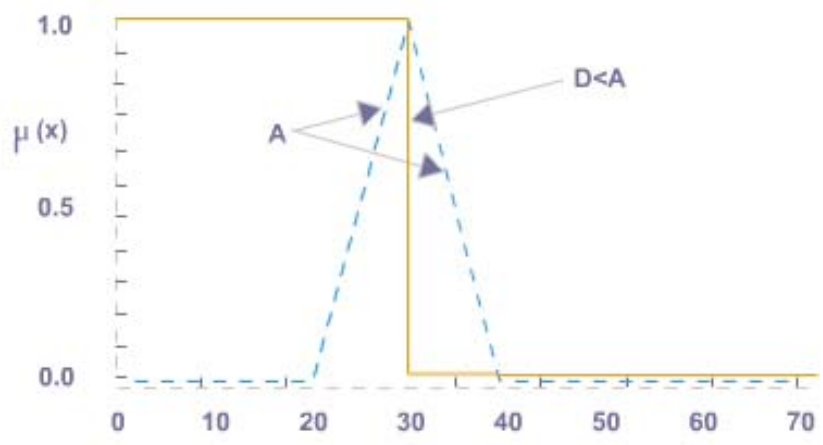


الشكل -12- العدد العائم **A**.

ثم نكون العدد العائم **C= NOT A** وهو مبين في الشكل -13-. ثم ننشئ عدداً عائماً **D** أقل من **A** بالمعنى البولياني **true-false**. قيمة **D(x)** تساوي الواحد حين يكون **A(x)** أقل من قيمته العظمى، **x** تبدأ من ∞- . وبعدها حين تزداد **x** تصبح **D(x)** تساوي الصفر. لنسمي القيمة **x** التي تصبح عندها **A(x)** عظمى بالقيمة **x(Amax)**. ينتج عن هذه العملية **D<A** كما في الشكل -14-



الشكل -13- العدد العائم **C= NOT A**.



الشكل -14- العدد العائم **D<A** بالمعنى البوليانى.

الفصل السادس

النظم الخبيرة العائمة

النظم الخبيرة العائمة

1. مقدمة

سنتحدث في هذا الفصل عن إجرائية المحاكمة العائمة؛ كيف نحصل على نتائج من قواعد ومعطيات، في حالات الشك والغموض والتناقضات. النتائج التي نحصل عليها في المراحل الأولى للمحاكمة يمكن أن تتغير أو تصبح غير صالحة أثناء سير إجرائية المحاكمة من مرحلة إلى المرحلة التي تليها.

تسمى عملية الوصول إلى النتائج من المعطيات الاستدلال **inference**، إذ نستنتج حقائق جديدة من القديمة. في فرضيات المنطق التقليدي، لدينا قيمتان فقط للحقيقة **true** و **false**، أما في المنطق العائم فلدينا الفرضيات وقيم الحقيقة التي تتجاوز القيمتين السابقتين في المنطق التقليدي. ولذلك، علينا أن نعرف كيف نحصل على قيم حقيقة جديدة من القديمة.

رأينا سابقاً مفهوم الإمكان **possibility**، الذي يهتم به الرياضيون، والذي يقيس إلى أي مدى تفشل المعطيات في دحض نتيجة. في العالم الحقيقي، نهتم بالحاجة **necessity** التي تقيس إلى أي مدى تدعم المعطيات نتيجة معينة. تختلف إجرائية المحاكمة التي تبني نتائج ضرورية عن تلك التي تبني نتائج ممكنة. على سبيل المثال، حين نستبدئ **initiate** قيم حقيقة ممكنة في غياب المعطيات فإننا نعطيها القيمة **1**، أما حين نستبدئ قيم حقيقة ضرورية في غياب المعطيات فإننا نعطيها القيمة **0**. من جهة أخرى، حين نتعامل مع الإمكان، تُعد مودس بوننس العائمة **fuzzy Modus Ponens** باستخدام معامل الاقتضاء أداة هامة للاستدلال، أما حين نتعامل مع الضرورة، فإننا نستخدم **t-norms** عوضاً عن معامل الاقتضاء.

من الآن فصاعداً، سنتعامل مع الضرورة وليس مع الإمكان، مالم نذكر صراحة عكس ذلك. حيث يمكن أن تتغير المعطيات والحقائق أثناء الاستدلال. يحدث هذا التغيير حين تُنفذ القواعد وتنجز التعليمات الناتجة **then part** هذا التعديل. علينا أن نعلم الظروف التي تسمح بتنفيذ هذه التعليمات. التي كنا نعتبرها سابقاً فرضيات أما الآن فهي تعليمات مباشرة.

2. أنواع الاستدلال العائم

ثمة ثلاثة أنواع للاستدلال العائم المعتمد على القواعد:

- المطرد المتزايد **monotonic**: وفيه يمكن لقيم الحقيقة للنتيجة أن تزداد ولكن لا تنقص.
- غير المطرد **non-monotonic**: وفيه يمكن لقيم الحقيقة للنتيجة أن تزداد أو تنقص.
- المطرد المتناقص **downward monotonic**: وفيه يمكن لقيم الحقيقة للنتيجة أن تنقص ولكن لا تزداد.

ونعرف أربع طرائق لتحديد قيم حقيقة النتيجة من الشرط **antecedents**، هي الموافقة لأنواع الاستدلال السابقة إضافة إلى المحاكمة التقريبية. وثمة ثلاثة خواص نرغب فيها، تتمتع بها أنواع الاستدلال الثلاثة، أما المحاكمة التقريبية فلا تتمتع بجميع هذه الخواص.

حين نقوم قيمة الحقيقة للشرط، ونرى أن الشرط صحيح بقدر كاف، نكون قادرين على تنفيذ النتيجة. تتضمن تعليمات النتيجة: الإنشاء **creation** والتعديل **modification** وإدخال المعطيات أو إخراجها **input or output data**، واتجاه تدفق البرنامج بقواعد مترفّعة **meta-rules**، وتعليمات التفلية **debugging** وغيرها. سنهتم فقط بالتعليمات التي تغير القيم أو تنشئها أو تغير قيم الحقيقة أو تنشئها.

3. طبيعة الاستدلال في النظم الخبيرة العائمة

يعتمد إمكان تعديل المعطيات على قيمة الحقيقة الموجودة وعلى قيمة الحقيقة للمعطيات الجديدة وعلى نوع الاستدلال. أما إنشاء المعطيات الجديدة فهو دائماً مسموح؛ والمسألة هي: ما مدى الثقة التي يمكن إسنادها لأي قيمة تحددها قيم المعطيات الجديدة؟

الشكل العام للقواعد هو:

IF (P) THEN (Q)

الشرط **P** هو فرضية عائمة، والنتيجة **Q** هي مجموعة تعليمات تُنفذ حين تُقدح القاعدة. يحدد الشرط مجموعة تأكيدات حول المعطيات. يمكن أن تكون هذه التأكيدات بسيطة (مثل وجود المعطيات أو أن قيمة الحقيقة لها صحيحة دائماً)، أو أقل بساطة مثل: قيمة الحقيقة للمعطيات صحيحة. يتطلب الشرط في غالب الأحيان مقارنة بين قيمة المعطيات أو قيمة الحقيقة لها، وتلك القيم الخاصة بحرفي **litteral** أو بمعطيات أخرى. وفي معظم الأوقات، تكون الفرضية مركبة

من عدة فرضيات بينها روابط مثل **AND, OR, NOT**. في جميع الحالات، ينتِج تقويم الشرط قيمة حقيقة وحيدة **single truth value**.

لدينا نوعان لتعديل المعطيات. يطبق النوع الأول على الواصفات ذات القيمة الوحيدة مثل الأعداد الصحيحة أو الحقيقية أو المحارف، وفيها يمكن تغيير المعطيات أو قيم الحقيقة لها. ويطبق النوع الثاني على الواصفات ذات القيم الوحيدة أو المتعددة، حيث يمكن فقط تعديل قيم الحقيقة؛ حين يسمح بتعديل قيم الحقيقة، يسمح أيضاً بتعديل الواصفات الوحيدة القيمة.

يمكن كتابة القاعدة التي تعدل قيم الحقيقة كما يلي:

IF (P) THEN (B'=B)

الشرط **P** كما في السابق، أما **B** فهي المعطيات الموجودة و **B'** هي المعطيات المراجعة **revised data**. وهدفنا إقرار إذا كان بالإمكان تعديل قيمة الحقيقة أم لا.

يمكن أيضاً أن يكون شكل القاعدة كما يلي:

IF (A'=A) THEN (B'=B)

إن **A** و **B** قيم محددة أولية، أما **A'** فهي قيمة جديدة نلاحظها، تختلف في غالب الأحيان عن القيمة **A**. إذا كانت **A** مجموعة عائمة، وجب تعريف **A'** على نفس الفضاء الكلي الذي تُعرف عليه **A**. وينطبق الكلام نفسه على المجموعتين **B** و **B'** (يجب أن يكونا معرفَين على الفضاء الكلي نفسه).

نسند إلى القاعدة نفسها قيمة حقيقة ولتكن **r**. (القيمة المغتفلة لـ **r** هي **1**.) تتغير الثقة بالشرط قبل تنفيذ تعليمات النتيجة بالاستعاضة عن **P** بـ **P AND r** (**AND** أو **min**). فيما يلي سنفترض دائماً أن قيمة **P** هي ناتج **AND** بين **P** و **r** ونفترض قيمة **r** تساوي **1**.

الاستدلال المطرد المتزايد monotonic inference

نفترض فيه أن قيمة الحقيقة لـ **B** موثوقة بأنها صحيحة، وأننا لم نقم بأي استنتاج خاطئ حتى الآن، وأي معطيات جديدة يمكنها فقط أن تزيد ثقتنا بأن قيمة **B** صالحة.

لنفترض مثلاً أن درجة عضوية **B** هي **0.8**، وأنه يجري حالياً قدح قاعدة ثقتنا بجزء الشرط منها **0.5**. لأننا نثق بدرجة عضوية **B** فإننا نهمل المعلومات الجديدة، وتفشل القاعدة، وتبقى درجة عضوية **B** هي **0.8**.

إن صيغة **B'** (قيمة الحقيقة الجديدة لـ **B**) في هذا الاستدلال هي:

B'=P OR B

حيث **P** هي قيمة الحقيقة لجزء الشرط، ويمكن أن تكون من الشكل **(A'=A)**، فإذا افترضنا أن **A** و **A'** معطيات وحيدة القيمة (أعداد صحيحة أو حقيقية أو محارف) بقيم حقيقة مختلفة فعندها:

P = truth value (A'=A) = max (A, A')

فإذا لم تكن المقادير **A** و **A'** قيماً سلمية، وإنما مجموعات عائمة منفصلة فإننا نكتب:

P = truth value (A'=A) = max (A(x), A'(x)), ∀x in A, A'

وبالمشابهة إذا لم تكن **B** و **B'** قيماً سلمية فإننا نكتب: **B'(x) = max (P, B(x))**

الاستدلال غير المطرد non-monotonic inference

هنا نفترض أن المعلومات الجديدة التي نحصل عليها من قدح قاعدة جديدة، هي أكثر موثوقية من معلومات موجودة سابقاً.

لنفترض مثلاً أن درجة عضوية **B** هي **0.8**، وأنه يجري حالياً قدح قاعدة ثقتنا بجزء الشرط منها **0.5**. لأننا نثق بالمعلومات الجديدة، فإن قيمة الحقيقة لـ **B** تنخفض إلى **0.5**. بالمقابل، لو كانت ثقتنا بجزء الشرط من القاعدة الجديدة **0.9** لارتفعت قيمة الحقيقة لـ **B** إلى **0.9**.

في حالة المعطيات الوحيدة القيمة، فإننا نكتب:

B'=P

حيث **P** هي الثقة بأن **(A'=A)**. أما إذا لم تكن **B** و **B'** مقادير وحيدة القيمة، فإن المحاكمة غير المطردة لاتفيد مالم يكن **A, A', B, B'** معرفة على الفضاء نفسه، وفي هذه الحالة نكتب:

$B_j'(x) = A'(x)$ AND $A(x)$

B_j' هي المفردات اللغوية الأعضاء في **B'**. لكن هذا الاستدلال ليس ذا فائدة عظيمة في النظم الخبيرة العائمة.

الاستدلال المطرد المتناقص downward monotonic inference

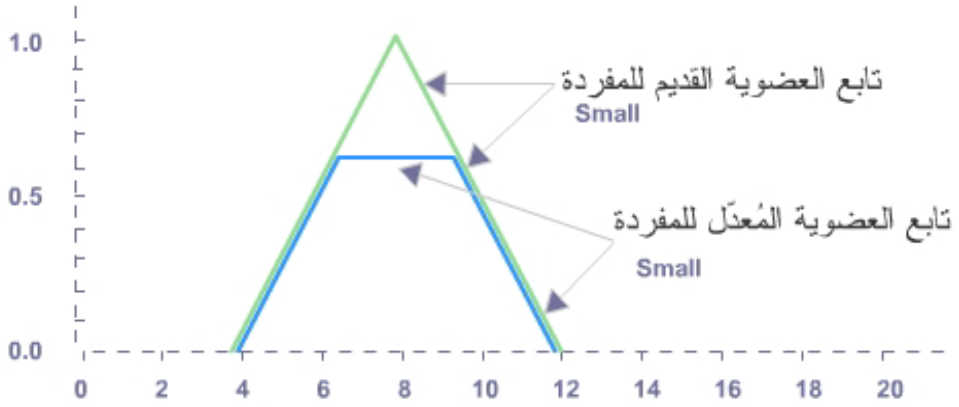
يمكن أن يكون هذا الاستدلال مفيداً إذا كانت قيمة الحقيقة السابقة لـ **B** تمثل حداً أعلى لما هو ممكن. وهو ضروري في حالة فك التعويم. لنفترض أن المتغير اللغوي لجزء النتيجة هو **B** وأن أعضاءه هي المفردات اللغوية **B_j**، يمكن أن يكون التصريح عن المعطيات والقاعدة كما يلي:

```
declare Data output fzset  (Small Medium Large);
rule IF (...) THEN  output  is Small;
```

حيث القاعدة هي:

If P_i then B is B_j

وعندها يكون: $B'_j = P \text{ AND } B_j$.



الشكل-1- تعديل تابع العضوية للمفردة **Small**، بقدح القاعدة **If A Then size is Small** والثقة بالقاعدة **0.6**

4. المحاكمة التقريبية approximate reasoning

ليس المقصود بالمحاكمة التقريبية هو المحاكمة على معطيات غير مؤكدة، وإنما المحاكمة باستخدام المنطق العائم. في المنطق التقليدي تشكل علاقة مودس بوننس عصب المحاكمة، وفيها نستنتج **B** من القاعدة **A→B** ومن **A**. يصبح لمودس بوننس في المنطق العائم الشكل التالي:

IF X is A THEN Y is B

From X=A'

Infer that Y=B'

A, A', B, B' مجموعات عائمة، وتوابع العضوية لهذه المجموعات هي غالباً أعداد عائمة.

ويحسب **B'** مباشرة من تركيب **A'** مع **A→B**، أي:

$$B' = A' \circ A \rightarrow B$$

لنأخذ المثال التالي:

$$A = \left\{ \frac{0.3}{x_1}, \frac{0.7}{x_2}, \frac{1.0}{x_3} \right\}$$

$$B = \left\{ \frac{0.5}{y_1}, \frac{1.0}{y_2}, \frac{0.6}{y_3} \right\}$$

ولتكن علاقة الاقتضاء هي المعرفة بالعلاقة: **T(x, y) = min (1, 1-x+y)** عندها تكون العلاقة **R(x,y)** لهذا الاقتضاء كما يلي:

$$R = \begin{bmatrix} 1 & 1 & 0.5 \\ 0.8 & 1 & 1 \\ 0.6 & 1 & 0.6 \end{bmatrix}$$

ونفترض أن

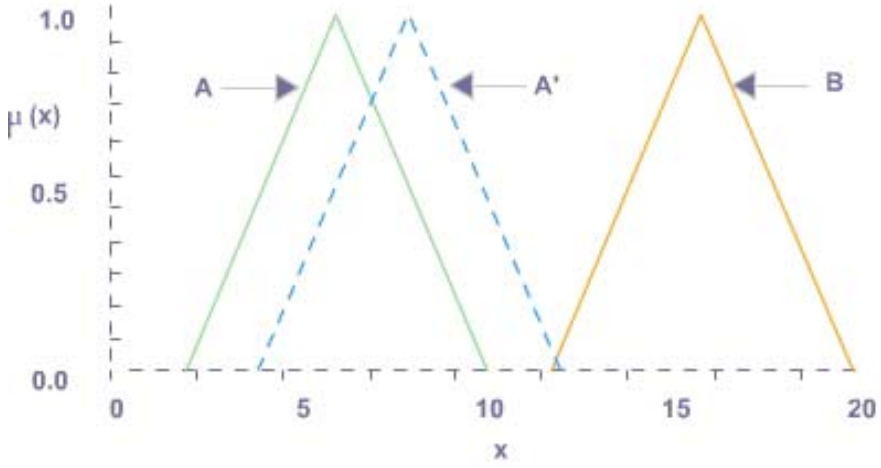
$$A' = \left\{ \frac{1.0}{x_1}, \frac{0.6}{x_2}, \frac{0.3}{x_3} \right\}$$

لنحسب الآن **B'** من **A' ° R** باستخدام **$T=T_m$** فنجد أن:

$$\left.\begin{aligned} B'(y_1) &= \max\{\min(1,1),\ \min(0.6,0.8),\ \min(0.3,0.6)\} = 1 \\ B'(y_2) &= \max\{\min(1,1),\ \min(0.6,1),\ \min(0.3,1)\} = 1 \\ B'(y_1) &= \max\{\min(1,0.5),\ \min(0.6,1),\ \min(0.3,0.6)\} = 0.6 \end{aligned}\right\} \Rightarrow B' = \left\{ \frac{1.0}{y_1}, \frac{1.0}{y_2}, \frac{0.6}{y_3} \right\}$$

مثال آخر: لتكن لدينا القاعدة: **if x is Big then y is slow**.

والمجموعات العائمة **A, B, A'** المعرفة بالشكل التالي:



الشكل -2- مثال لحساب محاكمة تقريبية

ولتكن العلاقة العائمة **R** التي تعرف الاقتضاء **(A→B)** هي العلاقة التالية (علاقة **Gaines-Rescher)**:

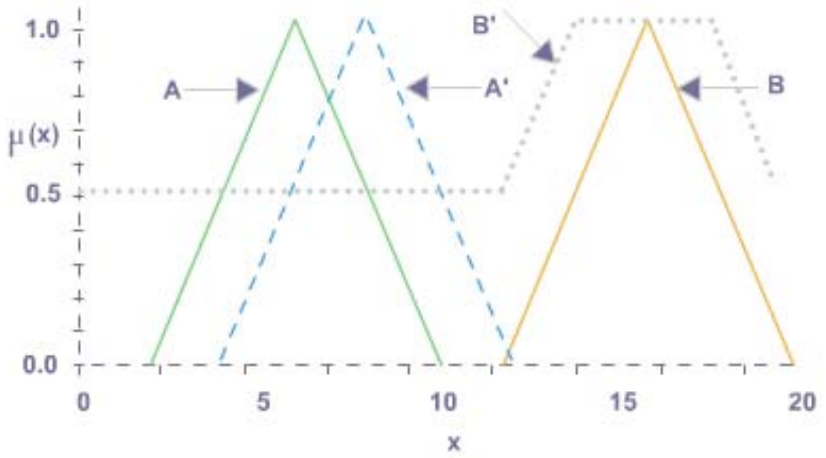
$$R(x,y) = tv(A(x) \to B(y)) = \begin{cases} 1 & if \quad tv(A(x)) \le tv(B(y)) \\ 0 & else \end{cases}$$

ولتكن علاقة التركيب التي تسمح بحساب تابع العضوية لـ **B'** هي العلاقة التالية:

$$B'(y) = \sup_x \{(A'(x),\ R(x,y))\}$$

حيث يشير **Sup_x** إلى أصغر عدد لا يتجاوز محددات التابع **arguments**. فإذا نفذنا الحسابات

على المجال **0≤x≤20** نحصل على **B'** المبينة بالشكل التالي:

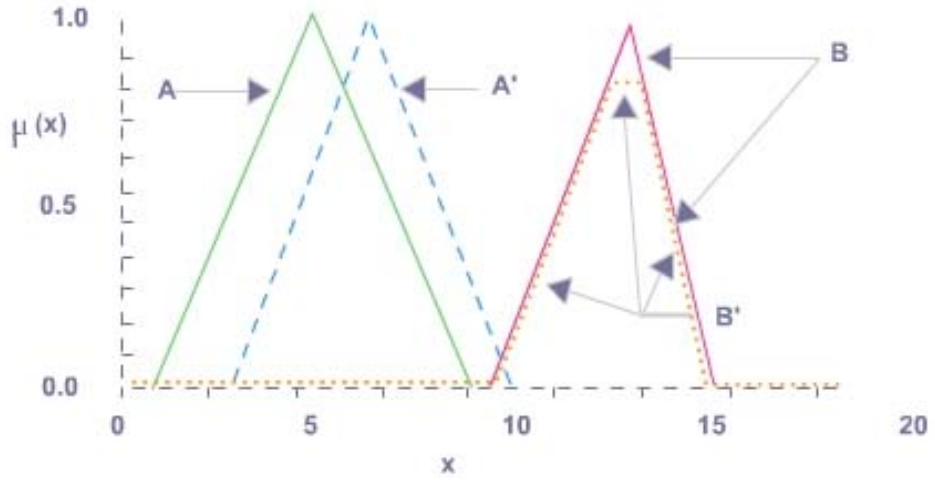


الشكل -3- ناتج تطبيق المحاكمة التقريبية على معطيات الشكل -2-

في غالب الأحيان لانستخدم هذه الحسابات في النظم العائمة لسلبياتها الكثيرة:

- قيم **B'** أكبر من **0.5** دائماً، لذلك لا يمكن فك تعويم تابع العضوية.
- عدم الاتساق (حين تكون **A=A'** كان يجب أن نجد **B=B'**).
- حين يكون **A∩A'=∅** وباستخدام T_m نحصل على **B'(y)=1**.

عوضاً عن علاقة الاقتضاء السابقة نستخدم **t-norm** (**AND** العائم)، فنحصل على الشكل التالي، وهو المستخدم فعلياً في النظم الخبيرة العائمة:



الشكل-4-: توابع العضوية **A, A', B, B'**. **B'** ناتج تركيب **A'** مع **(A AND B)**. على معطيات الشكل -2-

هذه العلاقة ليست علاقة اقتضاء، نسميها علاقة **Mamdani implication** لأنه هو من استخدمها

في التحكم العائم.

ولمعرفة الفرق بين المحاكمة التقريبية والمحاكمات الأخرى من القاعدة:

If X is A then Y is B

حيث **X, A, Y, B** معرفة على الفضاء نفسه، فإذا كانت **X** هي **A'** يجب حساب المجموعة العائمة **B'** وهي المجموعة **Y** التي توافق **A'**. وكانت **P** قيمة الحقيقة لجزء الشرط من القاعدة، نجد أن:

B'= P OR B في المحاكمة المطردة المتزايدة.

B'= P في المحاكمة غير المطردة.

B'= P AND B في المحاكمة المطردة المتناقصة.

B' = A' ° (A→B) في المحاكمة التقريبية.

خواص مفيدة:

نظن أن الخواص التالية مفيدة في كل من طرق المحاكمة الثلاثة التالية:

- المحاكمة المطردة المتزايدة.

$$1. B_j^{'} \geq B_j$$
$$2. If \;\; A' = NULL, then \;\; B' = B$$
$$3. If \;\; A' and \;\; A \;\; are \;\; disjo\,\text{int}, then \;\; B' = B$$

- المحاكمة غير المطردة.

$$1. B' = P = (A' = A)$$
$$2. If \;\; A' = NULL, then \;\; B' = NULL$$
$$3. If \;\; A' and \;\; A \;\; are \;\; disjo\,\text{int}, then \;\; B' = NULL$$

- المحاكمة المطردة المتناقصة.

$$1. B_j^{'} \leq B_j$$
$$2. If \;\; A' = NULL, then \;\; B' = NULL$$
$$3. If \;\; A' and \;\; A \;\; are \;\; disjo\,\text{int}, then \;\; B' = NULL$$

ونود تحقيق الخواص التالية:

- -الخاصة **A**، في المحاكمة المطردة المتزايدة: **B' = (A' = A) OR B**.
- -الخاصة **B**، في المحاكمة غير المطردة: **B' = (A' = A)**.
- -الخاصة **C**، في المحاكمة المطردة المتناقصة: **B' = (A' = A) AND B**. وهي مكافئة لطريقة **Mamdani**.
- -الخاصة **D**، في المحاكمة التقريبية: **$B' = A' \circ A \rightarrow B$**. في هذه الطريقة، الاقتضاء ينتج علاقة عائمة **$[A_i \rightarrow B_j]$** ويشير ° إلى معامل التركيب الذي يعرف بالعلاقة:

$B'_j = \max(\min(A'_i, A_i \rightarrow B_j))$

تحقق الطرق **A, B, C** الخواص المرغوبة لكل نوع من أنواع المحاكمة التي صممت من أجلها، وتفشل في تحقيق أي خواص أخرى. وتفشل المحاكمة التقريبية في تحقيق أي خاصة من خواص المحاكمات المطردة أو غير المطردة، وخاصة الخاصة الثالثة التي تقول إذا كانت **A** و **A'** منفصلتين بحيث **(A=A')** هي صفر، فمن الممكن أن تنتج **B'** هي واحد في كل مكان، وهذا مناقض للحدس !!

تفشل المحاكمة التقريبية في تحقيق الخواص المرغوبة لذلك لا تُستَخدم في النظم الخبيرة العائمة. ولكن أي الطرق يجب أن نطبق؟

المحاكمة المغتفلة **by default** متفائلة، نفترض أننا نثق بقيم الحقيقة لجزء النتيجة، وأن أي معلومات إضافية يمكنها فقط أن تجعل قيم الحقيقة صالحة أو تزيد من ثقتنا بها. وهذه هي المحاكمة المطردة المتزايدة. ولكن، أحياناً تأتينا معطيات جديدة نثق بها أعلى الثقة، في هذه الحالة يمكننا استخدام المحاكمة غير المطردة. أما إذا كنا جد متشائمين، فنقول أن المعطيات الجديدة يمكن أن تنقص قيمة الحقيقة، ونستخدم بالتالي المحاكمة المطردة المتناقصة. هذه الطريقة يمكن أن تُستَخدم فقط حين نعدل توابع العضوية قبل فك التعويم.

FLOPS تستخدم المحاكمة المطردة المتزايدة، **by default**، ويمكن أن نحول إلى المحاكمة غير المطردة باستخدام الأمر **TMSoff** ونعود إلى المحاكمة المطردة المتزايدة بالأمر **TMSon**.

الفصل السابع

دراسة حالات وأمثلة على النظم الخبيرة

1. نظم التخطيط

نظم التخطيط التقليدية

إن مفهوم التخطيط مألوف لدينا، فنحن باستمرار نضع خططاً تؤثر في حياتنا ونضبطها. قد يكون التخطيط طويل الأمد مثل تخطيط مهنة المستقبل، أو قصير الأمد مثل تخطيط وجبة الغداء. يمكن تعريف التخطيط كما يلي:

التخطيط هو توصيف مجموعة الأفعال والمؤثرات بترتيب مسبق لتحقيق هدف معين. وأحياناً نسمي التخطيط "المحاكمة على الأفعال".

التخطيط يشبه البرمجة ولكن، قد ينفذ البرنامج عدة مرات في حين قد تُنفذ الخطة مرة واحدة. من جهة أخرى، بيئة البرمجة معرفة مسبقاً في حين قد يتطلب التخطيط التعامل مع تغيرات غير متوقعة للمحيط.

من التطبيقات التي تحتاج إلى التخطيط أو المساعدة في التخطيط: قرارات إدارية، وتشكيلات المصانع، وتنظيم إجرائيات التصنيع، وتخطيط الأعمال والتنبؤ بها، وصنع القرارات العسكرية الاستراتيجية. بعض البرمجيات المعقدة يمكن أن تتمتع بتخطيط داخلي كأن تخطّط أفعالها هي. على سبيل المثال، تحتاج الربوطات أن تخطط حركاتها، في حين يمكن أن يستعمل التخطيط الداخلي للتأكيد على الحصول على حل مناسب خلال زمن مقبول. الجدولة هي تحصيص الموارد للإجرائيات ومزامنة هذه العمليات، وهي أحد أشكال التخطيط الصعبة والمعقدة.

نفترض في معظم نظم التخطيط أنه يمكن تمثيل العالم بلقطات **"snapshot"** تسمى حالات العالم. لذلك فالمحاكمة في مثل هذه الحالات تسمى المحاكمة المعتمدة على الحالات **state-based reasoning**. وهذا هو التخطيط التقليدي، إذ نفترض أننا بالتخطيط نعرف كيفية الانتقال من الحالة الابتدائية إلى الحالة الهدف، وهذا يتطلب توصيف كل من الحالة الابتدائية والهدف ومجموعة العمليات التي يمكن إنجازها في فضاء الحالات. يبين الشكل التالي تصنيف إجرائيات التخطيط بحسب السلم الزمني لعملية التخطيط.

الشكل **-1-** تصنيف إجرائيات التخطيط بحسب السلم الزمني لعملية التخطيط

إن فرضية اللقطات (الحالات) للعالم صالحة إذا لم يتغير العالم أثناء عملية التخطيط. لذلك، فإن التخطيط التقليدي غير مناسب لمحاكمة الإجرائيات المستمرة أو للتعامل مع الكوارث غير المتوقعة. إذا كان النظام قادراً على الاستجابة بسرعة كافية للتغيرات بحيث يخطط مجدداً وينفذ الخطة الجديدة، نقول إنه نظام تفاعلي **reactive** غير تقليدي.

لنأخذ على سبيل المثال، ذهابي من المنزل إلى الدكان، ثمة تغيرين واضحين على حالة العالم بعدها:

- أنا لم أعد في المنزل،
- أنا أصبحت في الدكان.

ثمة متغيرات أخرى قد لا تكون محسوسة مثل نقص كمية المادة في أسفل الحذاء نتيجة المشي. وثمة متغيرات لا علاقة لها بذهابي مثل تغير سعر البنزين في ذاك اليوم، وتغيرات لها علاقة بالسياق، فإذا كان الجو ماطراً وصلت إلى الدكان مبتلاً...

STRIPS

في التخطيط التقليدي، نفترض أن العالم لا يتغير من حالة إلى أخرى، مالم نذكر صراحة ما طرأ عليه من تغيرات، هذا الافتراض تبناه النظام **STRIPS** أقدم نظم التخطيط في الذكاء الصنعي.

بداية، جرى تطوير النظام ليسمح لروبوت أن يخطط طريقاً بين الغرف، وينقل صناديق على الطريق. ينمذَج العالم على أنه يتضمن أغراض ومؤثرات يمكن أن تُطبق على هذه الأغراض لتغير نموذج العالم. لكل مؤثر شروط تطبيق **preconditions** وآثار ناتجة **effects**. الطريقة المتبعة هي طريقة تحليل **means-ends** وهي تقيس المسافة بين الحالة الحالية وحالة الهدف

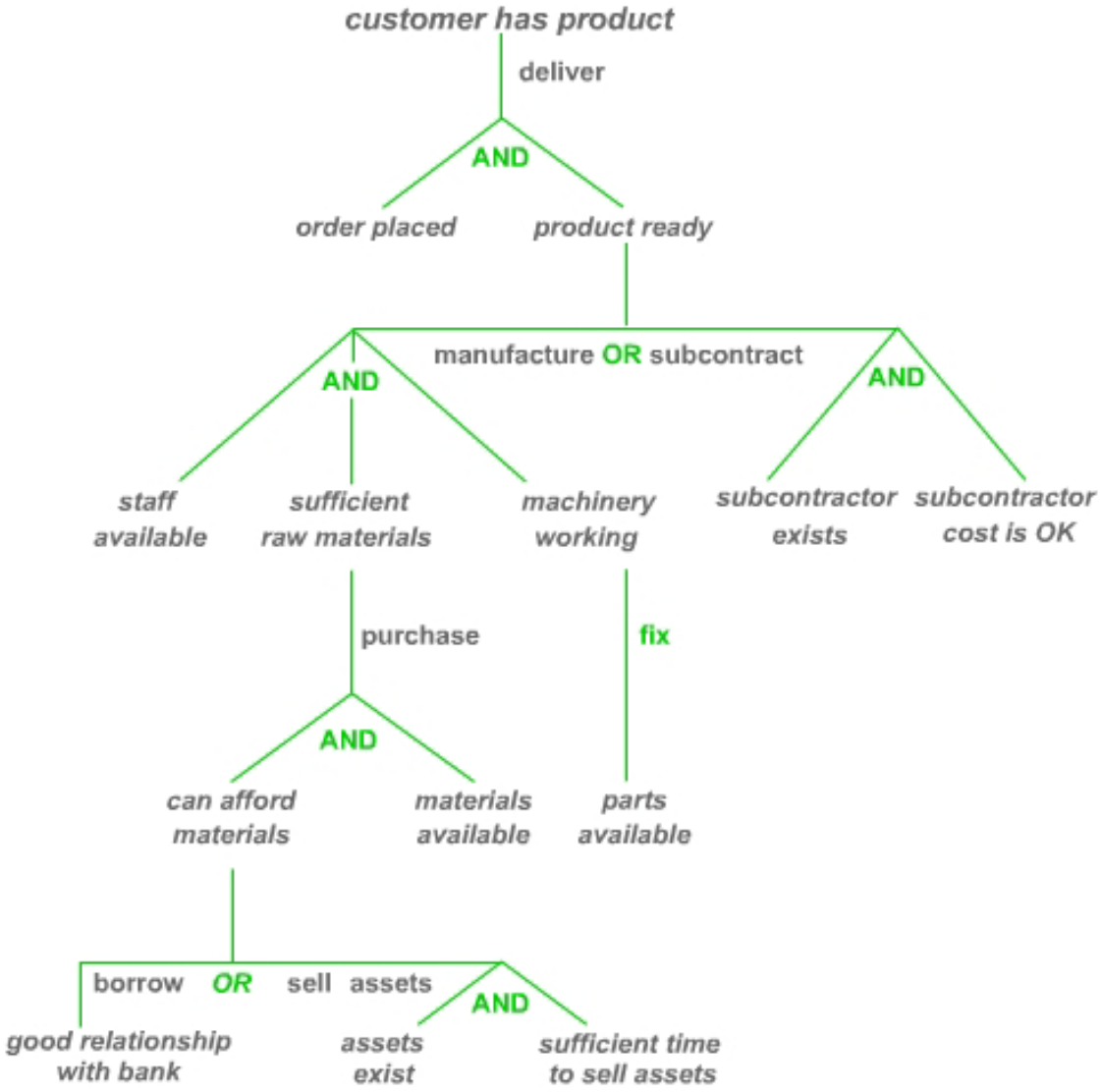
وتوجد المؤثر الذي يقلص هذه المسافة، وهكذا نبحث في المؤثرات إلى أن نصل إلى الهدف.

لنأخذ على سبيل المثال حالة أن يطلب الزبون منتجاً ما **product**، عندها يمكن أن نسلم المنتج **deliver** إذا كان جاهزاً وإذا طلبه الزبون **customer**. إذا لم يكن المنتج جاهزاً لدينا فيمكن أن نعهده لشخص آخر **subcontact** شرط أن يكون المتعهد **subcontractor** متاحاً وتكلفته أقل من سعر المنتج. وإلا فيمكن أن نصنّع المنتج **manufacture**، وهذا يمكن أن يتحقق إذا كان طاقمنا **staff** متاحاً والمواد الأولية **raw materials** متوفرة لدينا، والآلات **machinery** تعمل... يبين الشكل **-2-** المؤثرات لتوفير منتج مع شروط التطبيق والآثار. الأغراض في الشكل هي ما تحته خط.

الآثار	الشروط الأولية	المؤثر والغرض
حصل الزبون على الطلب	المنتج جاهز وقدّم الزبون الطلب	سلّم منتجاً
المنتج جاهز	جهة التعاقد متاحة وتكلفتها أخفض من سعر المنتج	تعاقد مع (مصنّع) المنتج
المنتج جاهز، موادنا الأولية أقل، آلاتنا تقترب من وقت الصيانة	العاملون متاحون ولدينا مواد أولية و الآلات تعمل	صنّع المنتج
لدينا مواداً أولية وسحب المال من حسابنا	يمكننا تأمين المواد الأولية وهي متاحة	اشترِ مواداً أولية
المال أُضيف إلى حسابنا	علاقة طيبة مع المصرف	اقترض المال
المال أُضيف إلى حسابنا	القيمة المضافة موجودة ولدينا الوقت الكافي لبيعها	بِع قيمة مضافة
الآلات تعمل، وجرت جدولة وقت الصيانة التالي	قطع التبديل متوفرة	صلّح الآلات

الشكل -2- المؤثرات لتوفير منتج مع شروط التطبيق والآثار. الأغراض هي ما تحته خط

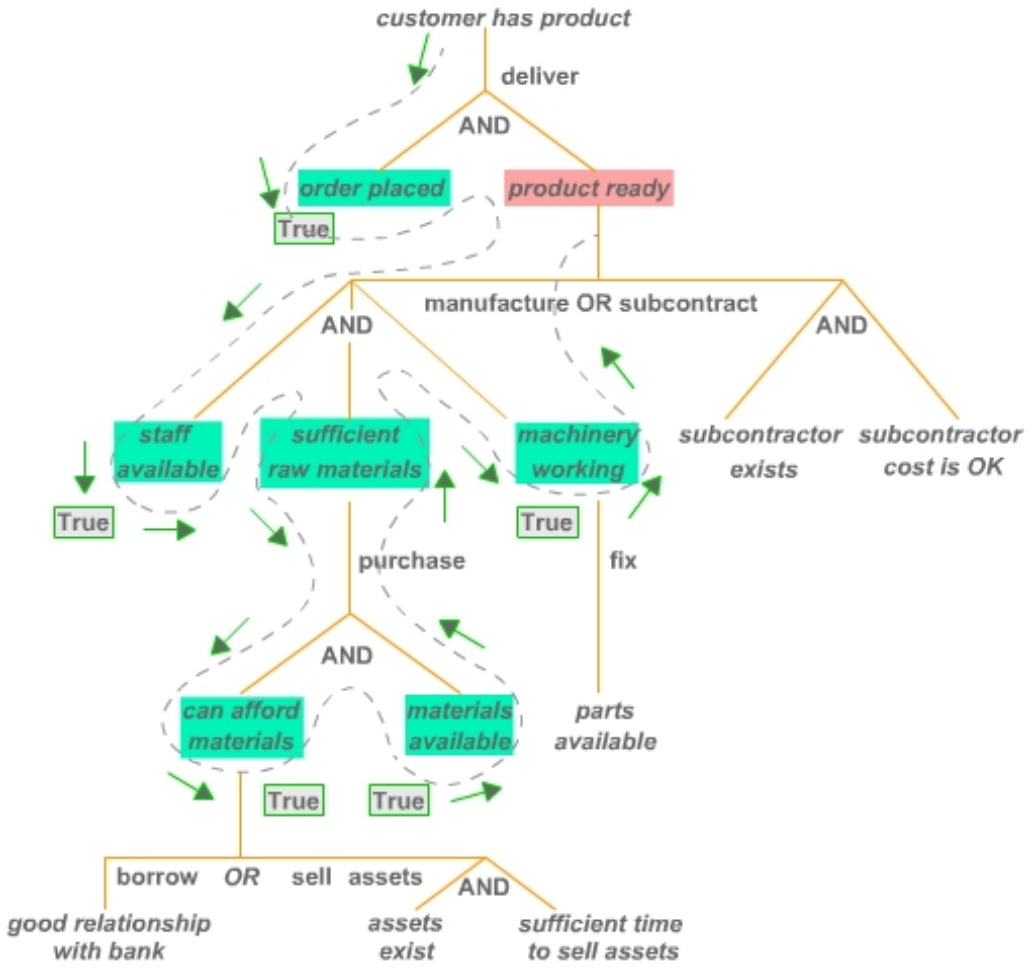
يمكن أن نمثل شجرة البحث عن المؤثرات وآثارها كما في الشكل -3-



الشكل -3- شجرة البحث عن الآثار والمؤثرات

فإذا كانت شروط البدء هي:

الزبون طلب المنتج- الزبون ليس لديه المنتج- الطاقم العمالي موجود- ليس لدينا مواد أولية- المواد الأولية موجودة عند المورد **supplier**- يمكننا شراء المواد الأولية- الآلات تعمل. فيمكن الوصول إلى الهدف عبر المسار المبين بالخطوط المنقّطة على الشكل -4- في شجرة البحث السابقة.



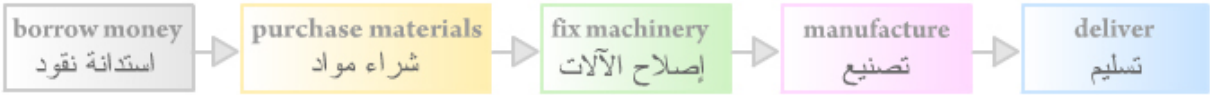
الشكل -4- الوصول إلى الهدف ضمن شروط البدء المذكورة

في التنفيذ الحقيقي لـ **STRIPS** نذكر جميع الآثار التي تنتج عن تطبيق مؤثر ما، وليس فقط الآثار التي لها علاقة بالوصول إلى الهدف. على سبيل المثال، شراء المواد الأولية يعطي أثراً ه علاقة بالهدف هو توفّر المواد الأولية، وأثراً آخر هو نقصان رصيدنا بالمصرف بعد شراء هذه المواد.

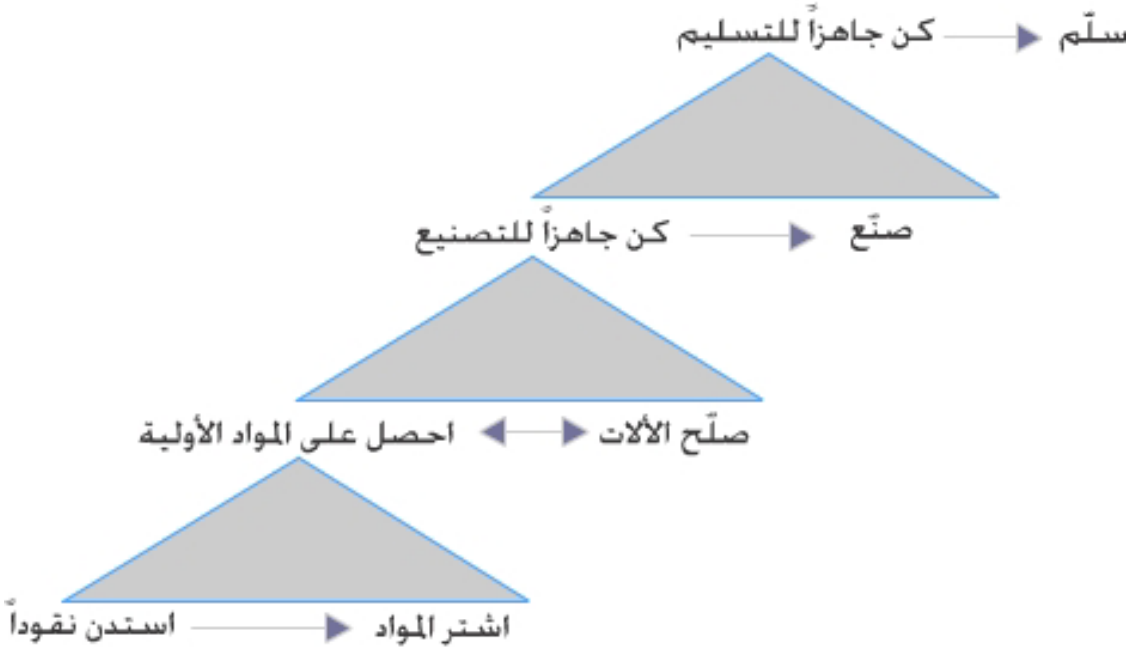
التخطيط التراتبي (الهرمي)

نظرياً جميع الخطط تراتبية **hierarchical** بطبيعتها، مع أننا لا نمثّلها هكذا في جميع نظم التخطيط. **STRIPS** على سبيل المثال ليس تراتبياً. إذ يمكن أن ينفّذ الهدف المطلوب باتباع الخطة التي تتضمن الخطوات التالية:

استدانة نقود- شراء مواد- إصلاح الآلات- تصنيع- تسليم.



في الأفعال السابقة، بعض الخطوات أساسية (مثل التصنيع)، في حين يشكّل البعض الآخر تفاصيل بسيطة (مثل شراء المواد). يبين الشكل -5- خطة تراتبية لتحقيق الهدف السابق.



الشكل -5- خطة تراتبية لتحقيق الهدف السابق

نسمي الأفعال التي لا تحتاج إلى تقسيمها إلى مجموعة أفعال أخرى أبسط منها، أفعالاً أولية، **primitive actions**. إن **ABSTRIPS** هو نظام **STRIPS** ولكن مع تجريد للخطة **abstraction** فهو بذلك تراتبي.

في التخطيط غير التراتبي يمكن أن نسبر مسارات غير فعالة أما التخطيط التراتبي فهو طريقة لتأجيل التعهدات (الضمانات) ريثما تتوفر معلومات حول صلاحية الفعل الذي يندرج في الخطة.

تأجيل الضمانات

رأينا أن التخطيط التراتبي هو طريقة لتأجيل التعهدات (الضمانات) ريثما تتوفر معلومات حول صلاحية الفعل. يمكننا تنفيذ ذلك بعدة طرق:

ترتيب جزئي للخطط

حين يوجد في الخطة أفعال غير مرتبة بعضها بالنسبة لبعض.

نقول عن نظام أنه خطي إذا كان يفترض أن ترتيب تنفيذ الأهداف المرتبطة (المعطوفة بعضها على بعض) **conjunctive** غير مهم. وإذا لم يعتمد النظام على هذا الافتراض فهو غير خطي.
إن توليد الخطط المرتبة جزئياً هي الجزء الأعم من النظم غير الخطية. حيث نؤجل ترتيب

الأفعال إلى أن تتوفر معلومات تساعدنا على ترتيب الأفعال.

استخدام متغيرات التخطيط

هنا نستخدم متغيرات كأن نقول على سبيل المثال: نشتري شيئاً ما. وحين نتبين ما علينا شراءه، أثناء سير الخطة، نسند قيمة ما نريد شراءه إلى المتغير "شيء ما".

الجدولة

الجدولة هي مسألة تخطيط، تتطلب تحصيص الموارد والزمن إلى المؤثرات المعروفة مسبقاً. أكثر مسائل الجدولة أهمية هي **job shop scheduling**، وهي حين يكون لدينا مجموعة آلات في مصنع أو جزء من مصنع، مع نظام نقل فيما بينها. قد يتطلب استكمال مهمة ما، إنتاج عدة أجزاء. ومسألة الجدولة هي إيجاد الجدول الزمني **schedule** لتصنيع مختلف القطع بهدف تحقيق مهمة ما. لذلك، فإن خرج مسألة الجدولة هو تحصيص الآلات (والموارد الأخرى) لكل إجرائية، وأزمنة البدء والانتهاء لكل عملية. يجب أن نؤكد على تنفيذ جميع المهام أثناء الأزمنة المطلوبة لها.

ثمة ثلاث طرق استُخدمت سابقاً للجدولة هي: التحليلية **analytical** والتكرارية **iterative** والتجريبية الكسبية **heuristic**. لم تنجح أي من هذه الطرق، وإنما ما حقق النجاح هو خليط من هذه الطرق الثلاث.

في النهج التحليلي نتطلب هيكلة المسألة في نموذج رياضي، وهذا بدوره يتطلب مجموعة من الافتراضات التي لا تكون واقعية دائماً. النهج التكراري يتطلب اختبار جميع الجداول الممكنة لاختيار الأنسب، وهو بدوره يتطلب كماً كبيراً من الحسابات. أما النهج الكسبي فيعتمد على استخدام قواعد خبير لقيادة الجدولة، ولكن هذه القواعد تكون في غالب الأحيان مناسبة لحالة معينة وليست عامة.

التحليل المعتمد على القيود

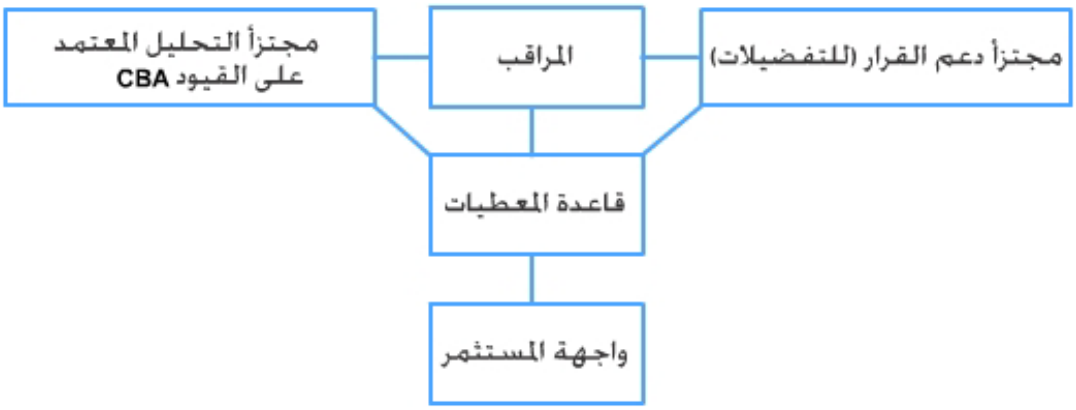
حين ننفذ أي عملية جدولة، ثمة قيود يجب احترامها، وتفضيلات نفضل أن نحققها. مثلاً يجب احترام موعد تسليم منتج ما، ويفضل تنفيذه على آلة من نوع معين. ثمة أربعة أنواع من القيود:

- قيود الإنتاج: يجب أن تكون كمية البضاعة جاهزة قبل موعد التسليم، وأن تحترم معايير الجودة، بزمن بدء لانبدأ قبله وزمن انتهاء لا نتأخر بعده.

- قيود التماسك التقاني: لا نبدأ بمنتج قبل دخوله خط الإنتاج، وأحياناً يوجد تسلسل مسبق للعمليات، وبعض المراحل تتطلب آلات معينة.
- قيود الموارد: لكل عملية حق النفاذ إلى قيود كافية (زمن على آلة)، وعدد الآلات المتاحة محدود. يمكن للآلة أن تعمل في لحظة معينة على منتج واحد معين في لحظة معينة. يجب أخذ مواعيد صيانة الآلات بالحسبان.
- قيود الإمكان: يجب ألا يتجاوز استخدام الآلة وطول خطوط انتظارها حدوداً معينة.

OPAL هو نظام جدولة يعتمد على القواعد، يحل مسألة جدولة الأعمال. وهو نظام تخطيط تقليدي يتضمن المجتزآت الخمسة التالية:

- قاعدة معطيات غرضية التوجه لتمثيل كينونات النظام مثل العمليات والعينات **lots** والموارد.
- مجتزأ التحليل المعتمد على القيود **CBA constrained based analysis** الذي يحسب آثار القيود الزمنية على سلسلة العمليات.
- مجتزأ دعم القرار ويضم قواعد لاختيار الجدول الزمني اعتماداً على الخبرة والتجربة.
- مراقب (مشرف) **supervisor** يتحكم بالاتصالات بين **CBA** ومجتزأ دعم القرار، وينشئ الجدول الزمني ليقدمه للمستثمر.
- مجتزأ واجهة المستثمر



الشكل-6- المجتزآت الرئيسية في نظام الجدولة **OPAL**.

إن أول مرحلة في مجتزأ التحليل المعتمد على القيود هي تحديد فيما إذا كان هناك تعارضات، وأين تحصل هذه التعارضات. نستخدم لهذه التعارضات مجموعات حرجة، نحدد فيها مجموعة العمليات التي تحتاج إلى موارد أكثر من المتاحة، ولكن يمكن حل التعارض إذا حذفنا أي عملية

من المجموعة. هناك ترتيب بين العمليات في المجموعات الحرجة. يجب أن تؤخذ القيود السابقة بالاعتبار عند الجدولة.

ملاحظة:

ثمة نظم يمكنها التعامل مع الكوارث مثل عطل آلة أو غياب عامل أو وصول طلبات مستعجلة أو الخطة الموضوعة غير جيدة. **SONIA** نظام تخطيط يمكنه تغيير خططه في هذه الحالات.

لذلك، فهو نظام تفاعلي. من الأفعال التي يمكن أن يتخذها: حذف بعض الإجرائيات أو تأجيلها لاحترام الجدول الزمني- إعادة جدولة استخدام أي فجوة زمنية في الخطة القديمة- إعادة تحصيص الموارد بين الإجرائيات- إعادة جدولة مهمة لتنتهي بعد الزمن المطلوب –تأجيل كامل الجدول الزمني.

2. نظم التحكم

للتحكم تطبيقات كثيرة في الصناعة والربوطية. وهدف مسألة التحكم هو تنفيذ خطة تصنيعية.

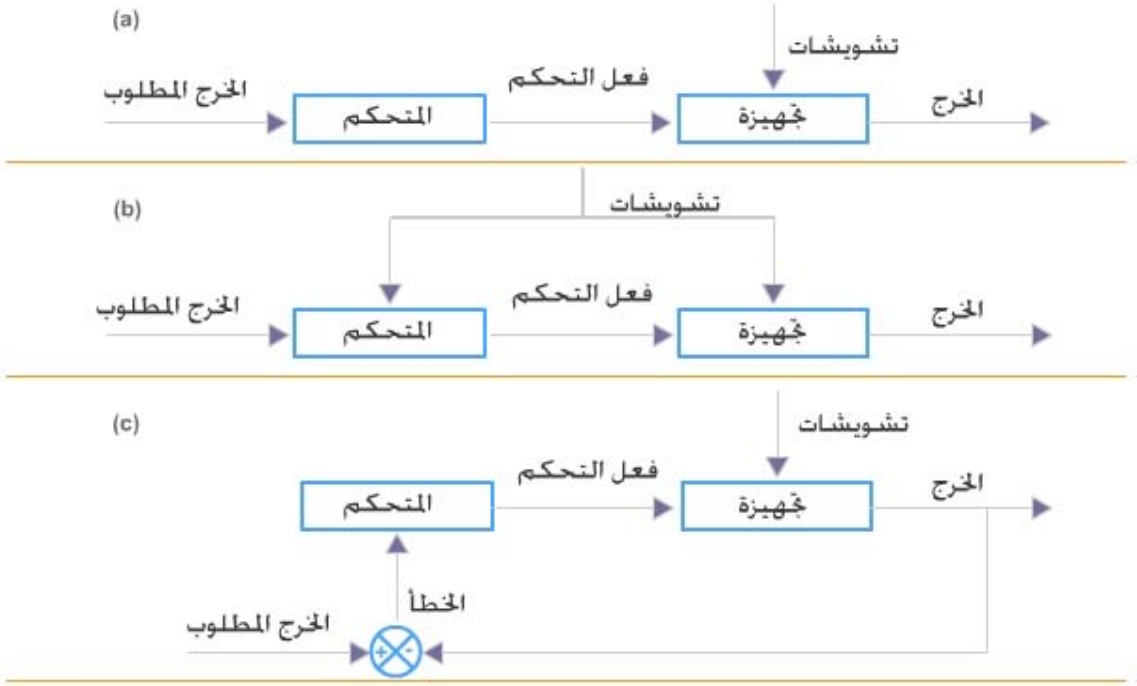
يتطلب التحكم تفسير معطيات المحسات وإدراك الأعطال والاستجابة لها. كذلك، يتطلب إعادة تخطيط عملية التصنيع في حالة التوقف أو الأحداث غير المتوقعة. نظم التحكم أعقد من النظم التي رأيناها سابقاً، فهي يمكن أن تتدخل وتغير بيئتها بإرسال أوامر إلى العتاديات أو بطلبها من مشغّل بشري التدخل للقيام ببعض الأفعال. مثال: التحكم بحرارة فرن.

التحكم المنخفض المستوى

ثمة التحكم في حلقة مفتوحة: يرسل المتحكم فعلاً تحكمياً للجهاز. المتحكم يمتلك نموذجاً للعلاقة بين فعل التحكم وسلوك الجهاز.

التحكم بتغذية أمامية **feedforward**: يأخذ بالاعتبار التشويش على الجهاز وذلك بقياس هذا التشويش **disturbance** وتغيير التحكم بناء على ذلك.

التحكم بتغذية راجعة **feedback**: نقيس خرج الجهاز ونقارنه بالقيمة المطلوبة ونستخدم الفرق أو الخطأ بينهما لضبط فعل التحكم.



الشكل -7- استراتيجيات التحكم الثلاث: a- حلقة مفتوحة b-تغذية أمامية c-تغذية خلفية (حلقة مغلقة)

في التحكم المنخفض المستوى نفترض أن بإمكاننا نمذجة الجهاز بمعادلة تفاضلية من الدرجة الأولى أو من الدرجة الثانية. ويمكن أن يكون نظام التحكم تماثلياً أو رقمياً. يشتهر نظام التحكم التناسبي التكاملي التفاضلي **PID**، الذي ينتج مقداراً متناسباً مع الخطأ ومع تكامله بالزمن ومع تفاضله وثمة تحكم **bang-bang** يتأرجح فعله بين حدوده الدنيا والعليا دون أن يسمح بالقيم البينية.

متطلبات التحكم المرتفع المستوى

قد يكون التحكم العالي المستوى مسألة معقدة تتعلق بقرارات حول أفعال يجب أن تُنفذ في أوقات محددة. وهذه قد تتطلب تحكماً تكيفياً **adaptive** أو آلياً **servo**. الهدف من التحكم التكيفي هو الحفاظ على حالة مستقرة وذلك بتكيف التحكم مع متغيرات المحيط. أما التحكم الآلي فهو مصمم لقيادة خرج الجهاز من قيمة ابتدائية إلى قيمة نهائية مرغوبة. مثال على التحكم العالي المستوى هو التحكم بإجرائيات صناعية، حيث يتطلب الأمر على سبيل المثال: اختيار المفاعلات والمواد الأولية، وسرعة أحزمة نقل المواد، ومعدلات تدفق سوائل ومواد صلبة وغازات، وتحكم بالحرارة والضغط...

وقد جرى تعريف متطلبات مفتاحية ستة لتحقيق التحكم العالي المستوى وبالزمن الحقيقي:

- إمكان اتخاذ قرارات تنفيذ أفعال ضمن قيود زمنية.
- التعامل مع أحداث غير متزامنة: للتعامل مع أحداث غير متوقعة.
- المحاكمة الزمنية.
- المحاكمة بوجود شك **uncertainty**.
- الإجرائيات المستمرة
- مصادر متعددة للمعلومات. (ضرورة التعامل مع نظام اللوح الأسود الذي يضم تسجيلاً لكل مايجري وما يجب أن يجري).

المحاكمة المقيدة زمنياً

لكي يعمل نظام التحكم بالزمن الحقيقي يجب أن يعمل بسرعة كافية، بحيث يستجيب إلى المعطيات الآتية بسرعة تساوي أو تفوق سرعة وصول هذه المعطيات.

إن جدولة منابع المعلومات موضوع تحكم داخلي ويجب أن يكون منفصلاً عن معرفة النطاق، أما القيود الزمنية فهي مرتبطة بالنطاق الذي نعمل ضمنه، ويجب ألا نضعها في محرك الاستدلال أو في برمجيات التحكم. هذا النهج اعتُمد في نظام **RESCU** الذي اعتمد المناهج التالية لتحقيق الزمن الحقيقي:

- وضع أولويات لمصادر المعلومات وللإجرائيات: المهام الهامة تُنفذ أثناء القيود الزمنية والمهام الأقل أهمية تُنفذ إذا سمح الوقت.
- التقريب: التأكد من الوصول إلى حل تقريبي خلال الزمن المتاح. ويجري تحسين الحل إن سمح الوقت. يمكن أن يضحي الحل التقريبي بالكمال أو بالدقة أو بالموثوقية (غياب الأدلة). يعرف بعضهم التقريب المعرف جيداً بأنه الذي يتمتع بالخواص التالية: يمكن التنبؤ بأثره على نوعية الحل وزمنه- نوعية الحل تتراجع بنعومة مع زيادة التقريب – غياب التمييزية لا يعني غياب الدقة.

التحكم العائم

وقد جرى مناقشته في الفصلين السابقين. نذكر هنا النظام **LINKman** الذي هو نظام تحكم عائم يطبق على التحكم بأفران الإسمنت وإجرائيات التصنيع الأخرى. والنظام **ARBS** وهو نظام لوح أسود للتحكم بتوضع البلاسما أثناء تصنيع الجذاذات الإلكترونية **electronic chips**، وهو يعتمد على قواعد عائمة.

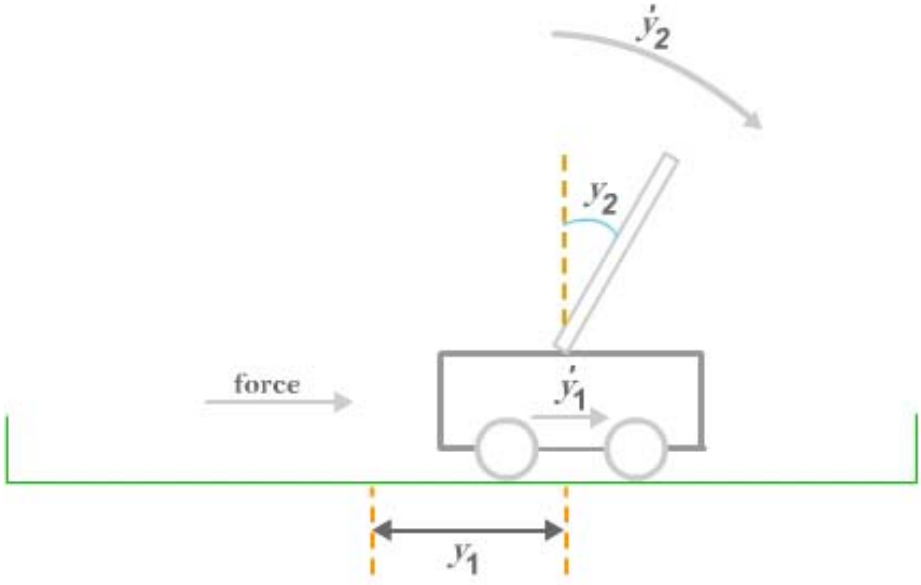
متحكم BOXES

ذكرنا سابقاً أن المتحكم يعمل إذا كان لديه نموذج للنظام الذي يتحكم فيه. خوارزميات الشبكات العصبونية و **BOXES** تسمح بتوليد مثل هذا النموذج من دون معرفة سابقة عن الآليات التي تحدث أثناء التحكم. هذا النهج مفيد في عدة حالات:

- إذا كان النظام معقداً بحيث لا يمكن نمذجته بدقة.
- ليست لدينا معلومات كافية عن النظام تسمح بنمذجته.
- لم نجد قواعد تحكم مقنعة.

يمكن تطبيق خوارزمية **BOXES** على التحكم التكيفي والآلي. مايلزمنا فقط هو: المدخلات (أفعال التحكم الممكنة) – المخرجات (تعرف حالة النظام في أي لحظة)- الحالة المرغوبة أو الحالة النهائية- القيود على المدخلات والمخرجات.

يجب تدريب متحكم **BOXES** لإرفاق التحكم المناسب مع كل صندوق **BOX** (حالة في فضاء الحالات)، والتحكم هو من النوع **bang-bang**. ولكي يتعلم النظام، يجب أن يتلقى قياساً لأدائه. تُستخدم خوارزمية **BOXES** للتحكم بالمفاعلات الحيوية **bioreactors** ولموازنة قطب على عربة متحركة (نواس مقلوب). ثمة خوارزمية مطورة عن **BOXES** هي النسخة العائمة منها. لتنعيم التحكم **bang-bang** بين حالتين.



الشكل-8- التحكم بنواس مقلوب. التحكم بالقوة لتوازن القضيب المقلوب.

التحكم الإحصائي بالإجرائيات

التحكم الإحصائي بالإجرائيات **statistical process control (SPC)** هو تقنية لمراقبة نوعية المنتجات أثناء تصنيعها. تُحدد الموسطات الحرجة وتُراقَب وتُضبط عملية التصنيع قبل تصنيع أي منتج لا يحقق المواصفات. تأتي أهمية هذا التحكم من تصغير عدد المنتجات المرفوضة في مرحلة التحكم بالجودة، وبذلك تزداد الإنتاجية وتتحسن.

تتطلب العملية مراقبة عينات من المنتجات المصنعة، وقياس موسطاتها الحرجة والاستدلال من هذه القياسات على توجهات الموسطات لكامل المنتجات.إن تجميع الإحصائيات ومعالجتها مهمة إجرائية، ويمكن استخدام كسبيات (تجريبيات) لتقفي أثر التوجهات في النظام. تقوم نظم إجرائية أو معتمدة على القواعد بعمليات المراقبة **monitoring**، وبحسب الإجرائيات يمكن أتمتة قرارات التحكم أيضاً. نهتم بالقيمة الوسطى والانحراف المعياري لموسطات العينات التي نجمع معطياتها.

المراجع

- **William SILER "Fuzzy expert systems and fuzzy reasoning", 2005m John Wikey & Sons, Inc.**
- **Kostas METAXIOTIS, Demitris ASKOUNIS and Konstantions NIKOLOPOULOS "Identifying the characteristics of successful expert systems: an empirical evaluation", Int, J. Information Technology and Management, Vol. 5, No. 1, 2006.**
- **-Adrian HOPGOOD, "Intelligent systems for engineers and scientists", second edition, 2005, CRC press.**