通用产生式系统语言模糊化扩充研究

康建初 （北京航空航天大学计算机系）

摘 要

Forward chaining production system, as a kind of computational model of knowledge representation,has been used to implement some of the most significant expert systems.However unfortunately,most of forward chaining production system languages make no provision for dealing with lexical imprecision. This paper briefly presents a fuzzy production system language model, which supports fuzzy matching between condition patterns of productions and facts in working memory, and its discussion is also focussed on the part played by the truth-values of fuzzy production instantiations in the conflict resolution of the language.

作为一种有用的知识表示形式，正向产生式系统结构已被人们广泛地用以建造各种专家系统。不过，大多数通用的正向产生式系统语言，如OPS5、YAPS等均未提供处理不精确语词（即模糊语词）的机制。另一方面，虽然一些有关模糊推理的实用模糊系统亦称为模糊产生式系统，那不过是因为它是数据驱动的。单从推理形式这一点看，它们被看成了正向产生式系统。然而，就其控制机制而言，模糊产生式系统与正向产生式系统有着很大的区别。

正向产生式系统的控制机制有这样的特征：在合一匹配阶段，一般有若干条规则的条件部份同时为事实库描述的外部世界的当前状态所满足，由此形成一个触发规则实例冲突集。然而，在系统的执

把由m维新证据E'推出结论C的可信度值确定为

$$CF(E')=\sum_{j=1}^{k}CF_j-\sum_{1\le j<l\le k}CF_j\cdot CF_l+\cdots+(-1)^{k-1}CF_1\cdot\cdots\cdot CF_k \qquad (8)$$

例 讨论位于某河流下游的城市八月份供水问题。证据有三方面：上游七月份的流量、下游八月份的气温和下游流域的降雨量。若气温高，则降雨量低，于是后两方面的证据不独立，应属于同一组。第一方面的证据独立于后两方面的证据，单独成一组。讨论的结论C是该市八月份缺水。

设由第一方面的证据推出C成立的可信度值$CF_1=0.6$，又设按上节的方法，第二，第三方面的证据推出结论C成立的可信度值$CF_2=0.5$

根据(8)，由三维新证据E'推出结论C的可信度值

$$CF(E')=CF_1+CF_2-CF_1\cdot CF_2$$
$$=0.6+0.5-0.6\times0.5=0.8$$

参考文献

〔1〕 Shafer, G., A Mathematical Theory of Evidence, Princeton University Press, 1976.

〔2〕 L. 约翰逊，专家系统技术指南，11.3.4，世界图书出版公司，1989.7.

〔3〕 Duda, R.O., Hart, P. E., Subjective Bayesian Methods for Rule-Based Inference System, AFIPS, 1976, 1075-1082.

〔4〕 魏宗舒等，概率论与数理统计教程，高等教育出版社，1986，117—118.

〔5〕 姚玉川等，知识系统，大连理工大学出版社，1988，194—201.

〔6〕 R. 福西斯等，专家系统原理和实例研究，中国铁道出版社，1989，64—66.

行阶段，根据各问题领域的具体规定，系统通常只执行冲突集中部份规则实例，或者极端地说，只执行集合中一条规则实例。这样，产生式系统都需要一个冲突解决算法，以便在它的规则选择阶段，能具体地确定冲突集中哪一部份或者哪一条规则实例应被优先执行。而一个模糊产生式系统[3]在经模糊匹配后，也会获得一个触发规则实例集。但是，因为它的实例集中每个实例的执行效果都相同，不同的只是各实例的执行强度，所以，在系统的一个执行周期内，实例集中任何一个触发实例均可作为启用实例。

本文试图以一个具体的通用正向产生式系统实验性语言——MUFL[6]为基础，探讨如何通过修改通用产生式系统语言的有关控制机制(如合一匹配，冲突解决策略等)达到使语言具有模糊产生式系统的某些功效（如识别和处理受限模糊语词)之目的。以下，我们称经模糊扩充后的MUFL为"FMUFL"。

一、关于模糊语词的处理

众所周知，模糊语词是模糊概念的语言表述形式。各问题领域涉及的模糊概念在产生式程序中都表现为具有不同语义的模糊语词。因此，一个通用的模糊产生式系统必须具有能分门别类地分析和处理这些模糊语词的能力。

根据Zadeh提出的语言变量和语言值的概念以及利用语言变量来分析自然语言语词结构的思想[4]，我们把可能涉及到的语言变量以及它们可取的语言值分成如下两类：

****第一类语言变量** 这类语言变量的论域是可以数量化的。变量所取的语言值均可以通过一个能够精确测量计算的数值变量x来描述，这个数值变量x叫做语言变量的基本变量，它在整个实数轴上或其某部份上取值，其取值范围就是论域。这类语言变量的系统结构可用图1说明。这类语言变量的例子还有许多。这些由语言值构成的集合分别对应不同的模糊概念系统。

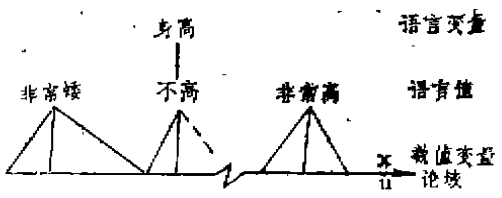

图1 第一类语言变量系统结构

****第二类语言变量** 这类语言变量的论域是不能数量化的，并且也没有可精确定义的基本变量x来刻画所取的语言值。人们往往是根据经验、印象、采取主观打分的方法，来近似地定量描述这类变量的语言值中各对象在性态方面的区别程度。这类变量的例子有"外貌"，"品行"…，它们所取的语言值分别为{美，丑，…}，{善良，凶残，…}。在FMUFL中，我们分别用I-type集合和可枚举的模糊集合来定义以上提及的这两类语言变量的语言值的语义。

所谓I-type集合[1]是指在一个连续无穷的论域上，利用线性插值方式表示的具有连续统的势的集合。例如，我们可以用I-type集合将语言变量"食物的价格"的一个语言值"昂贵"(expensive)定义如下：

table(expensive, food, [0,200],[[20, 0], [50, 0.5][70, 0.8], [100, 1]])，其中，区间[0, 200]代表论域，是由食物的价格构成的集合；"expensive"是该论域中一个模糊子集的标记；这个模糊子集被表示为一个序偶集：集合中各序偶的第一个元素为论域中的一个元素，本例中，它为价格；第二个元素是[0, 1]区间中的一个实数，指示第一个元素属于这个模糊集合的程度，换句话说，实数指示了论域中某客体与此模糊集合所表征的模糊概念之间的相容程度[5]。如果已知两个价格及其各自在此模糊集合中的隶属度，那么位于这两个价格之间的任一价格的隶属度都可用线性插值法求出。

可枚举模糊集合用来定义第二类语言变量的语言值的语义，实质上是由产生式系统程序中具有相同谓词的目标级事实组成。在表示这类模糊集合时，我们利用了隶属函数在模糊集合与模糊逻辑间所起的沟通作用。这里，我们假定各事实的谓词为某个可枚举模糊集合的标记，事实中的客体是论域中的

元素；客体满足谓词的程度，或者说，此事实成立的真值（用α表示，α∈〔0，1〕）代表论域中某对应元素在谓词指定的模糊集合中的隶属度。例如，语言值“饿”（hungry）可用以下可枚举的模糊集合定义成：

hungry(mary) with truth 0.8
hungry(tom) with truth 0.5
hungry(john)

由此定义可见，这类集合中各元素的隶属度是相对分配的。

为了避免为过多的语言值的语义下定义，我们为FMUFL提供了一个“语义关系转换装置”。这样，当一个语言值可以表示成另一个语言值的函数时，程序不必显式地给出第一个语言值的语义，而只需给出第二个语言值的语义以及两语言值间的语义关系。FMUFL可以通过“语义转换装置”求得第一个语言值的语义。FMUFL可以处理的语义关系为同阶和高阶的同义词、反义词。例如，对于前面定义的语言值“昂贵”（expensive），假定程序给出如下语义关系：

antonym(expensive, cheap).

则FMUFL认为语言值“便宜”（cheap）的含义应解释为“不贵”（not expensive）*)，反映其语义的I-type集合应为：

table (cheap, food, 〔0, 200〕, 〔〔20, 1〕, 〔50, 0.5〕, 〔70, 0.2〕, 〔100, 0〕〕)。

类似地，对于上面定义的语言值“hungry”，若程序给出它的高阶同义词“饿极了”(ravenous)如下：

modified-synonym(ravenous,very hungry)**)，

那么模糊语词“ravenous”被解释成“非常饿”（very hungry），FMUFL自动用下面的可枚举模糊集来表示其语义：

ravenous (mary) with truth 0.64
ravenous (tom) with truth 0.25
ravenous (john).

根据以上识别各类模糊语词义的方法。我们对通用产生式系统语言MUFL原有的合一匹配机制和基于假言规则的推理机制进行了修改，修改后的FMUFL在表现和处理产生式规则时，具有模糊产生式系统的以下特征：

(1) FMUFL允许产生式规则的条件部份出现表示模糊概念的语词，并且允许事实库中的事实含有模糊概念的模糊命题。这样，二者之间的匹配不必一定为严格的合一匹配，而可以是模糊的部份匹配。

例如 假设我们要写入程序的事实为“玛丽有些饿。”因为事实中出现的模糊词“饿”属于第二类语言变量的语言值，因此它可被加到事实库以谓词“hungry”为标记的可枚举模糊集合中。于是，事实库中有：hungry(mary) with truth0.8.这个事实表明玛丽饥饿的程度与已定义的模糊概念“hungry”的相容度为0.8。类似地，模糊命题“玛丽比较喜欢吃面包”，可表示成likes(mary, bread)with truth 0.7.加到事实库中以谓词“likes”为标记的模糊集合中去。FMUFL通过前面定义的语义关系，如modifiedsynonym (ravenous, very hungry)，将以上表示的模糊事实去与产生式规则的模糊模式条件，如，

ravenous (who),
likes (mary, what)

进行模糊匹配，并作模糊推理，求得这些条件在事实库描述的外部世界的当前状态下成立的真实程度，或者说，求得论域中有关元素在上述条件指定的模糊集合中的隶属度。如果规则的模式条件中出现的模糊语词是第一类语言变量的语言值，如

price (bread, food, expensive),
price (what, food, very cheap).

则模糊匹配和模糊推理的过程就稍复杂些。因为整个推理过程不仅要涉及到事实库中有关事实，模糊语词之间的语义关系，还要用到定义此类语词语义的I-type模糊集合。例如，匹配上面两个模糊模式条件，FMUFL需要对，如

事实：price (bread, food, 35)

*) 修饰词“not”定义为对原集合中各元素的隶属度“取补”，即not $A=1-A$。
**) 修饰词“very”定义为对原集合中各元素的隶属度求“平方”，即，very $A=(A)^2$。

I-type集：table (expensive,food,〔0, 200〕, 〔〔20, 0〕, 〔50, 0.5〕, 〔70, 0.8〕, 〔100, 1〕〕)

语义关系：antonym (expensive, cheap)

进行处理，才可求得此二模式条件成立的真值。

(2) 按照上述办法，FMUFL在求得某规则实例各条件成立的真值后，可进一步综合出该实例为事实库描述的当前状态所满足的总真值β($\beta\in$〔0, 1〕)，并且在执行某规则操作部份指定的各种操作时，它还可执行包括对事实库增、删模糊事实的操作。

例如，设有产生式规则如下：

when ravenous(P) and likes (P, F) and
price (F, food, fairly cheap)
then store should-buy (P, F) qualified.

该规则指出只要存在有满足规则中模糊条件的一对项 (P, F) (其中，P代表人，F代表食品名)，就要将这一对项 (P, F)，连同与其有关的参数加到事实库中以"should-buy"为标记的模糊集合中去。在此，有关参数是指这一对项 (P, F) 在集合should-buy中的隶属度，或者说是被插入模糊命题的真值。标志"qualified"指出这个真值应等于规则的条件部份中复合公式的真值。因此，若给定事实库内容为：

hungry (mary) with truth 0.8 (1)
likes (mary, bread) with truth 0.7 (2)
price (braed, food, 35) (3)
antonym (expensive, cheap) (4)
modified-synonym (ravenous, very hungry) (5)
table (expensive, food, 〔0,200〕, 〔〔20, 0〕, 〔50, 0.5〕, 〔70, 0.8〕, 〔100, 1〕〕) (6)

FMUFL将按照下述过程求出规则对应实例条件部份成立的真值，并将此实例：

when ravenous (mary) and
likes (mary, bread) and
price (bread, food, fairly cheap)
then store should-buy (mary, bread) qualified

送入冲突集：FMUFL首先把规则的条件部份中各个条件视为子目标，逐个地将它们与事实库中对应的模糊事实进行模糊匹配。由此分别获得这些条件在当前状态下成立的真值，即由 (1) 和 (5) 得：ravenous (mary) 的真值是0.64；由 (2) 得：likes (mary, bread) 的真值是0.7；由 (3), (6) 和 (4) 以及线性插值，可计算出：price (bread, food, cheap) 的真值为0.75；再由修饰词"fairly"*)的定义得：price (bread, food, fairly cheap) 的真值是0.87。其次，如果我们将逻辑与 ($\wedge$) 和逻辑或 ($\vee$) 分别解释为"取小" (min) 和"取大" (max)，FMUFL计算出上述规则条件部份为当前状态满足的程度为

$$0.64\wedge0.7\wedge0.87=0.64,$$

于是，在执行此规则的操作后，FMUFL在事实库中插入一个新的模糊事实：

should-buy (mary, bread) with truth 0.64。

至此，我们已经探讨了模糊产生式语言处理一条模糊产生式规则的技术，其中包括模糊概念的表示，规则条件与事实的模糊匹配，规则实例的真值计算等。然而，如前所述，在模糊产生式系统语言的一个执行周期中，同样会有多条模糊规则不同程度地为当前状态所满足。那么如何相应地扩充MUFL的冲突解决策略，使它能够从具有众多的模糊触发规则实例的冲突集中，作出合理的决策，即挑出本周期内的启用实例。就将在下面进行讨论。

二、关于冲突解决策略的模糊扩充

产生式系统的冲突解决策略实质上是一组触发实例选择原则。这组原则可以形象地比喻成一组按其网眼大小排列有序的筛子。于是，从冲突集中挑选启用规则实例的过程可对应地看成对触发实例逐层进行筛选的过程。只有最终通过所有筛子的触发实例（或实例子集）才能成为启用实例被执行。根据各自针对的问题领域不同，不同的产生式系统的冲突解决策略可能会有着一组不同的选择原则，并且原则应用的顺序可能也不尽相同。

冲突解决策略是产生式系统的一个重要组成部份，而且构成策略的基本内容（即选择原则以及其实施顺序）直接影响着一个系统的两点重要特性——系统的敏感性和稳定

*) 修饰词"fairly"定义为对原集合中各元素的隶属度求"平方根"，即$\text{fairly}A=(A)^{0.5}$。

性。这里，我们说一个系统是敏感的，是指此系统能对其工作的环境和外部世界状态的变化及时作出反响；我们说一个系统具有稳定性，是指系统能不受干扰地维持其原有操作的连续性。产生式系统的这两个特性是一对矛盾。我们认为，对于一个通用的产生式系统语言来说，在一般情况下，保证产生式系统程序在运行过程中的敏感性较保证其稳定性更为重要，因为前者可以使产生式系统更鲜明地表现出区别于其它计算模式的"数据驱动"的特征。

OPS5是一个出现较早的产生式系统语言。实践证明，OPS5为基于产生式规则的程序设计提供的通用方法和策略仍属最行之有效一类语言之列。MUFL的冲突解决策略与OPS5的策略相同。因此，我们要求扩充后的模糊产生式系统语言FMUFL的冲突解决策略必须与OPS5向上兼容，也就是说，如果在一产生式程序的运行中未涉及到模糊语词的话，FMUFL运用冲突解决策略的效果应与OPS5一样；但是一旦程序在问题求解中要求处理模糊语词，FMUFL的冲突解决策略应能适应由此引起的关于模糊规则实例选择的需要。下面，我们先简单地介绍MUFL原有的冲突解决策略（详见[6]），然后，在此基础上，进一步分析讨论有关策略的模糊扩充问题。

MUFL支持两种冲突解决策略：一是正则冲突解决策略，二是上下文限制冲突解决策略。前者适合于那些不具有明显结构，从而无法预先确定求解算法的问题领域；后者则适合于这样一些问题领域，即所要求解的问题可以进一步分解为一些相互之间有着明显先后求解顺序的子问题，但子问题内部却无明显结构。因为，在这后一类问题求解中，对所有子问题的求解工作可以按照一个子任务序列完成，所以可以说，上下文限制的冲突解决策略特别适合用来实现面向任务的程序设计思想[3]。

尽管这两种策略针对具有不同结构的问题领域，但它们包含的选择原则却大致相同。现在按它们逐个作用于冲突集的顺序分别规定如下：

*refractoriness原则　如果某规则的一个实例已在上一个执行周期被执行过，并且在后来的周期中，这条规则又为事实库中同一状态所满足而形成与上一次相同的实例，则此实例不应作为触发实例再一次进入冲突集。

* relative recency原则　如果某触发实例是通过某规则的条件与事实库中最新加入或最近修改过的事实合一匹配后获得的，则它可以保留在冲突集中。除此之外，其它的触发规则实例均应被抛弃。

* relative element specificity原则　当每两两比较触发实例时，如果其中某一实例对应的规则与事实库中事实合一的条件较多，则该实例可保留在冲突集中，而另一个实例应被抛弃。

* relative test specificity原则　条件部份具有较多的测试条件的规则实例被保留在冲突集中，而其它的实例则应被抛弃。

* arbitrary choice原则　冲突集中任一触发实例均可视为启用实例。这是个最弱原则。它实质上对冲突集中的触发实例作无条件地取舍，因此，只有在对冲突集用遍上面提及的全部原则而无法获得唯一的启用实例时，才使用这个原则。

模糊产生式系统语言FMUFL的冲突解决策略除了需要以上五条原则外，还需要增加其它新的原则，以便使它能够根据模糊推理中求得的标志多触发实例成立的真值β($\beta\in[0,1]$)的大小，合理地筛选触发规则实例。我们称这个新增加的原则为真值（truth value）原则。下面，我们讨论这个原则及其在策略中的位置，或者说，讨论它与其它原则相对应用顺序关系。

依处理时机不同，一个实例的真值既可视为绝对真值，又可视为相对真值。绝对真值指一实例未进入冲突集之前的真值，需用

一阈值度量，此阈值可以规定成0.5，或由程序员给出。我们可以规定凡是真值小于阈值的实例不得作为触发规则实例进入冲突集。这一规定称为"绝对真值"(absoluted truth value)原则，可以保证产生式程序的效率以及推理结论的准确性。显然，这个所谓"绝对真值"原则应置于冲突解决策略 refractoriness原则之前，以便有效地阻止其真值低于阈值的实例进入冲突集。一旦实例进入冲突集后，实例的真值可视为相对真值。这时，我们需要创建一个称之为"相对真值"(relative truth value)的实例选择原则，以保证具有（相对）最高真值的实例被保留在冲突集中，而其它均被抛弃。另一方面，我们将面临如何确定"相对真值"原则在原MUFL策略中应处的相对位置问题。换句话说，对于图2中给出的五个有序原则，下面我们将分析究竟应将"相对真值"原则置于何处，才能使产生式程序运行正常，所获结果合理。

a⇒
refractoriness
b⇒
relative recency
c⇒
relative element specificity
d⇒
relative test specificity
e⇒
arbitrary choice
f⇒

图2 "相对真值"原则可能出现的位置

(1) 假设置于a处 我们知道，设置refractoriness原则是为了有效地防止产生式程序的运行进入死循环状态。这类原则能使产生式系统表现出对其工作状态和工作环境的敏感性，它应是策略中的首要原则。因此，"相对真值"原则不能置于a处。

(2) 假设置于f处 因为arbitrary choice原则是一条最弱的原则，从本质上说，它采用的是一种随机地取舍实例的办法。而"相对真值"原则作为一个有条件地挑选实例的方法理应置于"arbitrary choice"原则之前而非f处。

(3) 假设置于b处 既然一个触发规则实例的真值反映的是实例对应规则的条件部份与事实库描述的当前状态的相容程度，可以说，"相对真值"原则的设置可使产生式系统表现出对事实库描述的外部世界状态的敏感性。然而，relative recency原则的敏感性更高一级，可以使运行中的程序能注意到外部世界状态的任何突然变化，从而可及时作出反应。利用这个原则，可以保障产生式系统的"数据驱动"特性。因此，在相应的模糊产生式系统的冲突解决策略中，我们理应优先考虑"recency"原则而不是"相对真值"原则。显然位置b也不适合。

(4) 假设置于c处 前面我们已说明：MUFL的上下文限制冲突解决策略特别适合于面向任务一类的程序设计。MUFL规定程序的每个产生式规则条件部份的第一个条件必须用来指明本规则所属的子任务名，而事实库必须以事实的形式唯一地给出当前任务名。MUFL利用不断更换当前任务名的办法来使产生式程序能连续地从一个子任务转向另一个子任务执行。显然，模糊产生式系统语言FMUFL也应保留这种面向任务的程序设计风格。为此，我们将用下面例子来说明在FMUFL的冲突解决策略中，为什么relative element, specificity原则必须在"相对真值"原则前面加以应用。设

事实库：

price (pepsi, food, 35).
selected ([bread, jam, potato-chips]).
task-is (add-cheap-extras).

规则库：

R1: when task-is (add-cheap-extras) and
selected (List) and
potato-chips in List and
untrue (pepsi in List) and
price (pepsi, food cheap)

then make Newlist=pepsi plus List and replace selected(List) by selected (Newlist).

R2: when task-is (add-cheap-extras) then remove task-is (add-cheap-extras).

因为规则2是要从事实库中撤掉当前任务"add-cheap-extras"的标记，从而终止此任务，所以从直观上看，规则2应在规则1完成之后再执行。"relative element"原则能保证这一点。现在我们假设首先考虑的是"相对真值"原则。在经过一系列的模糊匹配后（见第1节），规则1实例条件部份的真值是0.75，规则2条件部份的真值为1。根据"相对真值"原则，系统将取规则2作为启用规则。这样，上面程序永远不能执行规则1实例指定的操作，即在购物单上添上便宜的食品。只有当食物绝对便宜时，即食品的价格在"便宜"这一模糊集中的隶属度为1时，程度才能根据下一个选择原则relative element specificity执行规则1的实例，把食品加到购物单中。若果真如此，模糊产生式系统语言就名存实亡了，因为它已经完全丧失了处理不精确语词的能力。由此可见，在模糊产生式系统语言的冲突解决策略中，relative element specificity原则应该优先于"相对真值"原则，所以，"相对真值"原则不能置于c处。

(5) 假设置于d处 我们曾说过，在设计通用产生式系统语言的冲突解决策略时，一般应首先保证系统对外部世界状态的敏感性特征，其次才考虑系统的抗干扰特征。比较"相对真值"原则与"relative test specificity"原则，我们发现二者对产生式系统分别有以下影响：由于触发规则实例的条件部份的真值是通过模糊匹配由事实库中事实提供并经语言的模糊推理机制综合后得到的，所以从"相对真值"原则选择具有最高真值的实例这一点来看，该原则与事实库描述的外部世界状态有关。我们似乎可以说，它确实在真值方面保证了产生式系统对外部世界状态的敏感性。而relative test specificity原则只是根据产生式规则条件部份所含条件的多少来筛选规则对应的实例，它与外部世界的内容及其变化无关。因此，该规则作用的效果似乎只能反映产生式系统的抗干扰特性。综上所述，我们是否可以断定应将"相对真值"原则置于"lelative test specificity"原则之前？其实不然，上面(4)中例子就是一个反例。在这个例子中，规则1所需测试的条件多于规则2中条件，而前者条件部份的真值则小于后者条件部份的真值。若优先考虑"相对真值"原则，那么程序会同样失去规则1的模糊特性。因此，"相对真值"原则不应置于d处。

(6) 应置于何处？ 综上所述，我们最终将"相对真值"原则置于了e处，理由是：模糊产生式系统语言的冲突解决策略只不过是人们在选择启用规则实例方面的经验总结，是缺省时为程序员提供的一个一般性工具，它不应绝对地操纵基于规则的产生式程序的运行流程。如果程序员想要人为地调整一下程序流程，使某条规则的实例优先执行，那么他可以通过relative test specificity原则人为地给这条规则设置较多的测试条件。"相对真值"原则并无此调整程序流程的功能，因为在产生式程序设计阶段，程序员通常难以预测某规则实例的真值。况且事实的真值是外部世界的客观反映，程序员不可能提高触发规则实例的真值，使某规则实例优先执行。综上所述，我们最终按以下顺序排列模糊产生式系统语言FMUFL冲突解决策略中的触发实例筛选原则，它们沿箭头指向逐个地作用于触发规则实例冲突集。

absoluted truth value principle.
↓
refractoriness principle.
↓
relative recency principle.
↓
relative element specificity principle
↓
relative test specificity principle.
↓
relative truth value principle.
↓
abitrary choice principle.

（参考文献略）