Package ‘kst’

October 24, 2022

Version 0.5-4
Date 2022-10-24
Title Knowledge Space Theory
Description Knowledge space theory by Doignon and Falmagne (1999)
<doi:10.1007/978-3-642-58625-5> is a set- and order-theoretical framework, which proposes mathematical formalisms to operationalize knowledge structures in a particular domain. The 'kst' package provides basic functionalities to generate, handle, and manipulate knowledge structures and knowledge spaces.
License GPL (>= 2)
Depends R (>= 4.1.0), proxy, relations (>= 0.6-7), sets (>= 1.0-17)
Suggests Rgraphviz
Author Christina Stahl [aut],
David Meyer [aut],
Cord Hockemeyer [aut, cre]
Maintainer Cord Hockemeyer <cord.hockemeyer@uni-graz.at>
URL https://homepage.uni-graz.at/en/cord.hockemeyer/
NeedsCompilation no
Repository CRAN
Encoding UTF-8
Date/Publication 2022-10-24 13:52:37 UTC

R topics documented:

as.binaryMatrix ......................................................... 2
as.famset ................................................................. 3
as.relation ................................................................. 4
closure ................................................................. 5
kassess ............................................................... 6
katoms ................................................................. 7
kbase ................................................................. 8
as.binaryMatrix

Matrix Representation of Knowledge Structures

Description
Computes the matrix representation of a knowledge structure.

Usage
as.binaryMatrix(x)

Arguments
x An R object of class kstructure (or kspace).

Details
as.binaryMatrix takes an arbitrary knowledge structure in set representation and computes its matrix form.

Value
An R object of class matrix.

See Also
as.famset, kspace, kstructure

Examples
kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))

as.binaryMatrix(kst)
as.famset

Convert a binary matrix to a family of sets

Description

Create a set of sets from a binary matrix where each row of the matrix is taken as one set.

Usage

as.famset(m, as.letters = TRUE)

Arguments

m       A binary matrix.
as.letters        logical, should the elements of the sets be letters or numbers?

Details

as.famset takes a binary matrix and converts it to a family (i.e. set) of sets where each row of the matrix represents one set and a "1" in row i and column j means that element j is contained in set i.

If as.letters is TRUE the elements of the sets are letters, otherwise numbers. However, if the matrix has colnames, these are taken as names for the elements of the sets taking precedence over the as.letters parameter.

If the matrix contains the same row multiple times it is contained only once in the resulting family of sets.

Value

An R object of class set containing sets..

See Also

as.binaryMatrix, set

Examples

m <- matrix(c(1, 0, 0, 1, 1, 0), nrow = 2, ncol = 3)

m

as.famset(m)

as.famset(m, as.letters = FALSE)
Description

Computes the surmise relation of knowledge structures.

Usage

```r
## S3 method for class 'kstructure'
as.relation(x, ...)
## S3 method for class 'kbase'
as.relation(x, ...)
## S3 method for class 'kfamset'
as.relation(x, ...)
```

Arguments

- `x` An R object of class `kstructure`, `kbase`, or `kfamset`.
- `...` Currently not used.

Details

`as.relation` takes an arbitrary knowledge structure and computes the surmise relation of the corresponding quasi-ordinal knowledge space. Antisymmetric (and transitive) surmise relations may then be plotted as a Hasse diagram.

Value

An R object of class `relation`.

References


See Also

- `kstructure`, `kbase`, `kfamset`, `relation`, `plot`

Examples

```r
kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
                      set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
as.relation(kst)
```
**Description**

Computes the closure of knowledge structures.

**Usage**

```r
## S3 method for class 'kstructure'
closure(x, operation=c("union", "intersection"),...)
## S3 method for class 'kbase'
closure(x, operation=c("union", "intersection"),...)
## S3 method for class 'kfamset'
closure(x, operation=c("union", "intersection"),...)
```

**Arguments**

- `x` : An R object of class `kstructure`, `kbase`, or `kfamset`.
- `operation` : The set operation under which the closure is computed ("union" or "intersection").
- `...` : Other arguments to be passed to methods.

**Details**

The closure method for objects of class `kstructure`, `kbase`, or `kfamset` performs the closure of a knowledge structure, base, or family of sets by computing the "union", "intersection", "complement", or symmetric difference of any two knowledge states. "union" is also used as a basis for the `kspace` function.

**Value**

An R object of the same class as `x` where each subset represents one knowledge state of the resulting knowledge structure.

**Note**

The implementation of union is more efficient than the one in sets.

**References**


**See Also**

`kstructure`, `kspace`, `kbase`, `kfamset`, `closure`
Examples

```r
tst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
closure(tst, operation="union")
```

### Assess Individuals

Assigns individuals to their corresponding knowledge states.

#### Usage

```r
kassess(x, rpatterns=NULL, method="deterministic")
```

#### Arguments

- **x**: An R object of class `kstructure`.
- **rpatterns**: A binary data frame or matrix where each row specifies the response pattern of one individual to the set of domain problems in `x`.
- **method**: The desired assessment method. Currently only "deterministic" assessment is implemented.

#### Details

`kassess` assigns individuals to their corresponding knowledge state in a knowledge structure.

Assessing individuals based on a "deterministic" procedure starts by determining a domain problem `a`, which is contained in approximately half of the available knowledge states. If the individual being assessed has successfully solved the respective domain problem `a`, all knowledge states that do not contain domain problem `a` are removed from the set of potential knowledge states of the individual. If, on the other hand, the individual has not solved the respective domain problem `a`, all knowledge states that do contain domain problem `a` are removed from the set of potential knowledge states of the individual. From the remaining knowledge states a domain problem `b`, which again is contained in approximately half of the still available knowledge states, is selected. If the individual has successfully solved the respective domain problem `b`, all knowledge states that do not contain domain problem `b` are removed from the set of potential knowledge states of the individual. If, on the other hand, the individual has solved the respective domain problem `b`, all knowledge states that do contain domain problem `b` are removed from the set of potential knowledge states of the individual. This procedure is repeated until only one knowledge state is left. This is the knowledge state the individual is currently located in.

#### Value

A list where each element represents the knowledge state of one respondent.
**katoms**

**References**


**See Also**

`kstructure`

**Examples**

```r
# deterministic assessment
kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
                     set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
rp <- data.frame(a=c(1,1,0,1,1,0,1,0,0,0), b=c(0,1,0,1,0,1,0,1,0,0),
                  c=c(0,0,0,1,1,0,1,0,1,0), d=c(0,0,1,1,1,0,0,0,1), e=c(0,0,1,1,1,0,0,0,0))
kassess(kst, rpatterns=rp, method="deterministic")
```

**atom**s of knowledge structures

**Description**

Computes atoms of knowledge structures.

**Usage**

```r
katoms(x, items)
```

**Arguments**

- `x` An R object of class `kstructure` or `kbase`.
- `items` A set of items for which atoms are computed.

**Details**

For any item \( q \) of the knowledge domain \( Q \), an atom at \( q \) is a minimal knowledge state containing \( q \), where minimal refers to the fact that the respective knowledge state is not the union of any other knowledge states.

**Value**

A list where each element represents the atom(s) of one item specified in `items`.

**References**


**See Also**

`kstructure`, `set`
Examples

```r
kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
                    set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
katoms(kst, items=set("a","b","c"))
```

---

**kbase**

*base of a Knowledge Space*

---

**Description**

Computes the base of a knowledge space.

**Usage**

```r
kbase(x)
```

**Arguments**

- `x` An R object of class *kspace*.

**Details**

A *base* for a knowledge space is a minimal family of knowledge states spanning the knowledge space, i.e., the base includes the minimal states sufficient to reconstruct the full knowledge space. A knowledge structure has a base only if it is a knowledge space.

**Value**

A *kbase*, i.e. a set of sets where each subset represents one knowledge state of the base.

**References**


**See Also**

- *kspace*, *kstructure*, *set*

**Examples**

```r
kst <- kspace(kstructure(set(set("a"), set("a","b"), set("a","c"), set("a","b","d","e"),
                         set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
kbase(kst)
```
**kdomain**

*Domain of Knowledge Structures or Bases*

**Description**

Computes the domain of knowledge structures or bases.

**Usage**

```r
cdomain(x)
```

**Arguments**

- `x` An R object of class `kstructure`, `kbase`, or `kfamset`.

**Details**

A *domain* is a set of questions or items representing a field of knowledge.

**Value**

A *set* of items, each representing one question of the knowledge domain.

**References**


**See Also**

`kstructure`, `kbase`, `kfamset`, `set`

**Examples**

```r
kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
  set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
kdomain(kst)
```
**kfringe**  
*Fringes of Knowledge States*

**Description**  
Computes the fringe of a knowledge state.

**Usage**  

```r
kfringe(kst, state)
kfringe_inner(kst, state)
kfringe_outer(kst, state)
```

**Arguments**  

- **kst**: An R object of class `kstructure`.
- **state**: An R object of class `set`.

**Details**  
The fringe determines the symmetric difference between a given knowledge state and its neighboring states. It is divided into inner and outer fringe. The inner fringe contains the fringe items which are element of the knowledge state. They have probably been recently learned. The outer fringe contains those fringe items which are no element of the knowledge state. For these items, all prerequisites are fulfilled, i.e. the learner is ready to learn them now.

**Value**  
A set containing the fringe of state. If state is NULL then a list containing the fringes of all knowledge states is returned.

**References**  

**See Also**  
`kneighbourhood`, `kstructure`, `set`

**Examples**  

```r
kst <- kstructure(set(set(), set("c"), set("a","b"), set("b","c"), set("c","d"), set("d","e"), set("a","b","c"), set("b","c","d"), set("c","d","e"), set("a","b","c","d"), set("a","b","d","e"), set("b","c","d","e"), set("a","b","c","d","e")))
```
kneighbourhood

# fringe
kfringe(kst, set("c","d","e"))

kneighbourhood  Neighbourhood of Knowledge States

Description

Computes the neighbourhood of a knowledge state.

Usage

kneighbourhood(kst, state)

Arguments

kst  An R object of class kstructure.
state  An R object of class set.

Details

The neighbourhood of a knowledge state is the set of all those states which have a symmetric seu difference of 1.

Value

A set of sets containing the neighbourhood of state

References


See Also

kfringe, kstructure, set

Examples

kst <- kstructure(set(set()), set("c"), set("a","b"), set("b","c"),
set("c","d"), set("d","e"), set("a","b","c"), set("b","c","d"),
set("c","d","e"), set("a","b","c","d"), set("a","b","d","e"),
set("b","c","d","e"), set("a","b","c","d","e"))

# inner fringe
kneighbourhood(kst, set("c","d","e"))
knneighbourhood

Neighbourhood of Knowledge States

Description

Computes the neighbourhood of a knowledge state.

Usage

knneighbourhood(kst, state, distance)

Arguments

kst An R object of class kstructure.
state An R object of class set.
distance An integer specifying the size of the neighbourhood

Details

The n-neighbourhood of a knowledge state is the set of all those states which have a symmetric seu difference of not more than n.

Value

A set of sets containing the n-neighbourhood of state

References


See Also

kneighbourhood, kfringe, kstructure, set

Examples

kst <- kstructure(set(set()), set("c"), set("a","b"), set("b","c"),
set("c","d"), set("d","e"), set("a","b","c"), set("b","c","d"),
set("c","d","e"), set("a","b","c","d"), set("a","b","d","e"),
set("b","c","d","e"), set("a","b","c","d","e"))

knneighbourhood(kst, set("c","d","e"), 2)
knotions

Notions of Knowledge Structures or Bases

Description

Computes notions of knowledge structures or bases.

Usage

knotions(x)

Arguments

x

An R object of class kstructure or kbase.

Details

A notion is a set of items always jointly contained in some knowledge states. Consequently, these items carry the same information and may therefore be considered equivalent. A knowledge structure where each notion contains only one item is considered discriminative.

Value

A set of sets, each representing one notion of the knowledge structure.

References


See Also

reduction.kstructure, kstructure, set

Examples

kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
knotions(kst)
**Description**

Creates a knowledge structure from a surmise relation or knowledge states.

**Usage**

```r
kstructure(x)
```

**Arguments**

- `x`: Either an endorelation (see `relation`) representing a surmise relation, or a set of sets each representing one knowledge state.

**Details**

The most basic assumption of knowledge space theory is that every knowledge domain can be represented in terms of a set of domain problems $Q$. Moreover, knowledge space theory assumes dependencies between these domain problems in that knowledge of a given domain problem or a subset of problems may be a prerequisite for knowledge of another, more difficult or complex domain problem. These prerequisite relations are realized by surmise relations, which create a quasi-order between different domain problems. One advantage of these surmise relations is that they reduce the quantity of all possible solution patterns to a more manageable amount of knowledge states. Each of these knowledge states represents the subset of domain problems an individual is capable of solving. The collection of all knowledge states captures the organization of the domain and is referred to as knowledge structure.

`kstructure` takes an endorelation representing a surmise relation or a set of sets each representing one knowledge state (e.g., one clause of a surmise system) and returns the corresponding knowledge structure. A knowledge structure always contains the empty set and $Q$.

`kfamset` does essentially the same but without ensuring that the empty set and $Q$ are included.

**Value**

An `R` object of class `kstructure`.

**Note**

Note that by default the quotes indicate the fact that the items are represented by characters. For displaying purposes, these quotes may be turned off by setting respective set options (see `options`).

**References**

See Also

relation, set, options

Examples

# An endorelation representing a surmise relation
kst <- endorelation(graph=set(tuple(1,1), tuple(2,2), tuple(3,3),
    tuple(4,4), tuple(2,1), tuple(3,1), tuple(4,1),
    tuple(3,2), tuple(4,2)))
kstructure(kst)
# A set of sets representing knowledge states (e.g., clauses of a surmise system)
kst <- set(set("a"), set("a","b"), set("a","c"), set("d","e"), set("a","b","d","e"),
    set("a","c","d","e"), set("a","b","c","d","e"))
kstructure(kst)
# Turning off the quotes for displaying purposes
sets_options("quote",FALSE)
kfamset(kst)

kstructure_is_wellgraded

Well-Gradedness of Knowledge Structures

Description

Tests for the well-gradedness of knowledge structures.

Usage

kstructure_is_wellgraded(x)

Arguments

x An R object of class kstructure.

Details

A knowledge structure is considered well-graded if any two of its states are connected by a bounded path, i.e., each knowledge state (except the state for the full set of domain problems Q) has at least one immediate successor state that comprises the same domain items plus exactly one and each knowledge state (except the empty set {} has at least one predecessor state that contains the same domain items with the exception of exactly one.

kstructure_is_wellgraded takes an arbitrary knowledge structure and tests for its well-gradedness.

Value

A logical value.
\textbf{References}


\textbf{See Also}

\texttt{kstructure}

\textbf{Examples}

\begin{verbatim}
  kst <- kstructure(set(set(), set("a"), set("b"), set("c"), set("a","b"),
                      set("b","c"), set("a","b","c")))
  kstructure_is_wellgraded(kst)

  kst <- kstructure(set(set(), set("a"), set("b"), set("c"), set("a","b"),
                      set("a","b","c")))
  kstructure_is_wellgraded(kst)
\end{verbatim}

\section*{\texttt{ktrace} \hspace{1em} \textit{Trace of Knowledge Structures}}

\textbf{Description}

Computes the trace of knowledge structures.

\textbf{Usage}

\begin{verbatim}
  ktrace(x, items)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \textbf{x} \hspace{1em} An \texttt{R} object of class \texttt{kstructure}.
  \item \textbf{items} \hspace{1em} A set of items for which the trace is computed.
\end{itemize}

\textbf{Details}

The \textit{trace} of a knowledge structure $K$ on a set $A$ is the substructure of the knowledge structure $K$ on the set $A$, i.e., the substructure resulting from restricting the knowledge structure $K$ to the items specified in $A$.

\textbf{Value}

An \texttt{R} object of class \texttt{kstructure} where each element represents one knowledge state of the knowledge structure on the item specified in \texttt{items}.

\textbf{References}

kvalidate

See Also

kstructure

Examples

kst <- kstructure(set("a"), set("a","b"), set("a","c"), set("d","e"),
     set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e"))
ktrace(kst, items=set("c","d","e"))

---

kvalidate Validate Prerequisite Relations or Knowledge Structures

Description

Validates prerequisite relations or knowledge structures

Usage

kvalidate(x, rpatterns=NULL, method=c("gamma","percent","VC","DI","DA"))

Arguments

x An R object of class kstructure.

rpatterns A binary data frame or matrix where each row specifies the response pattern of
           one individual to the set of domain problems in x.

method The desired validation method (see details).

Details

kvalidate calculates validity coefficients for prerequisite relations and knowledge structures.

The \( \gamma \)-Index (method "gamma") validates the prerequisite relation underlying a knowledge structure
and assumes that not every response pattern is represented by a prerequisite relation. For this purpose
it compares the number of response patterns that are represented by a prerequisite relation (i.e.,
concordant pairs) with the number of response patterns that are not represented by a prerequisite
relation (i.e., discordant pairs). Formally, the \( \gamma \)-Index is defined as

\[
\gamma = \frac{N_c - N_d}{N_c + N_d}
\]

where \( N_c \) is the number of concordant pairs and \( N_d \) the number of discordant pairs. Generally, a
positive \( \gamma \)-value supports the validity of prerequisite relations.

The validation method "percent" likewise validates prerequisite relations and assumes that more
difficult or complex domain problems are solved less frequently than less difficult or complex domain problems. For this purpose it calculates the relative solution frequency for each of the domain problems in \( Q \).
The Violational Coefficient (method "VC") also validates prerequisite relations. For this purpose, the number of violations (i.e., the earlier mentioned discordant pairs) against a prerequisite relation are calculated. Formally, the VC is defined as

$$VC = \frac{1}{n(|S| - m)} \sum_{x,y} v_{xy}$$

where $n$ denotes the number of response vectors, $|S|$ refers to the number of pairs in the relation, $m$ denotes the number of items, and $v_{xy}$ again refers to the number of discordant pairs. Generally, a low VC supports the validity of prerequisite relations.

In contrast to the other three indices, the Discrepancy Index (method "DI" and the Distance Agreement Coefficient (method "DA") validate the resulting knowledge structure. The Discrepancy Index is the average distance between the response patterns and the knowledge structure

$$DI = \frac{1}{n} \sum_{r \in R} \min_{K \in K} d(r, K)$$

where $d$ is the symmetric set difference. Generally, a lower DI value indicates a better fit between a knowledge structure and a set of response patterns.

The Distance Agreement Coefficient compares the average symmetric distance between the knowledge structure and response patterns (referred to as $ddat$) to the average symmetric distance between the knowledge structure and the power set of response patterns (referred to as $dpot$). By calculating the ratio of $ddat$ and $dpot$, the DA is determined. Generally, a lower DA-value indicates a better fit between a knowledge structure and a set of response patterns. Please note that the $ddat$ value is equal to the $DI$ index. The DA coefficient is insofar a further development of the DI index as it takes into account the sizes of the domain and the knowledge structure and thus makes the DA values better comparable.

Value

Depending on the desired assessment method, a data frame with results for each domain problem (method "percent"), or a list (methods "gamma", "VC", "DI" and "DA") with the following components:

- **gamma** The gamma-value.
- **nc** Number of concordant pairs.
- **nd** Number of discordant pairs.

for the "gamma" method,

- **vc** The VC-value.
- **nd** Number of discordant pairs.

for the "VC" method,

- **di** The DI-value.
- **di_dist** The distance table for DI.

for the "DI" method, and
ddat  The ddat-value.
ddat_dist  The distance table for ddat.
dpot  The dpot-value.
dpot_dist  The distance table for dpot.
DA  The Distance Agreement Coefficient.

for the "DA" method.

References


See Also

* kstructure

Examples

```r
kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
rp <- data.frame(a=c(1,1,0,1,1,1,0,0,0), b=c(0,1,0,1,0,1,1,0,0),
c=c(0,0,0,1,1,0,1,0,0), d=c(0,0,1,1,0,0,0,1), e=c(0,0,1,1,1,0,0,0))

# Gamma Index
kvalidate(kst, rpatterns=rp, method="gamma")

# Percent
kvalidate(kst, rpatterns=rp, method="percent")

# Violational Coefficient
kvalidate(kst, rpatterns=rp, method="VC")

# Discrepancy Index
kvalidate(kst, rpatterns=rp, method="DI")

# Distance Agreement Coefficient
kvalidate(kst, rpatterns=rp, method="DA")
```
Learning Paths in a Knowledge Structure

Description

Computes learning paths in a knowledge structure.

Usage

lpath(x)

Arguments

x

An \texttt{R} object of class \texttt{kstructure}.

Details

A learning path in a knowledge structure is a maximal sequence of knowledge states, which allows learners to gradually traverse a knowledge structure from the empty set \(\emptyset\) (or any other bottom state) to the full set of domain problems \(Q\). Mathematically, it is represented as a set of states.

lpath takes an arbitrary knowledge structure and computes all possible learning paths in the respective knowledge structure.

Value

A list where each element represents one learning path.

References


See Also

kstructure

Examples

```r
kst <- kstructure(set(set(), set("a"), set("b"), set("a","b"),
set("a","d"), set("b","c"), set("a","b","c"), set("a","b","d"),
set("b","c","d"), set("a","b","c","d"), set("a","b","c","d","e")))
lpath(kst)
```
Description

Tests for the gradation property of learning paths.

Usage

lpath_is_gradation(x)

Arguments

x  A list of learning paths.

Details

A learning path is considered a gradation if each state in a learning path differs from its predecessor and/or successor state by a single item/notion.

lpath_is_gradation takes an arbitrary list of learning paths and tests for their gradation property.

Value

A list of logical values where each element represents one learning path.

References


See Also

kstructure, lpath

Examples

kst <- kstructure(set(set(), set("c"), set("a","b"), set("b","c"),
                     set("c","d"), set("d","e"), set("a","b","c"), set("b","c","d"),
                     set("c","d","e"), set("a","b","c","d"), set("a","b","d","e"),
                     set("b","c","d","e"), set("a","b","c","d","e")))
lp <- lpath(kst)
lpath_is_gradation(lp)
Description

Plots a Hasse diagram of a family of sets

Usage

```r
## S3 method for class 'kstructure'
plot(x, ...)
## S3 method for class 'kbase'
plot(x, ...)
## S3 method for class 'kfamset'
plot(x, ...)
```

Arguments

- `x` An R object of class `kstructure`, `kbase`, or `kfamset`.
- `...` Other arguments to be passed to methods.

Details

`plot` takes an arbitrary family of sets and plots a Hasse diagram.

References


See Also

`kstructure`, `kbase`, `kfamset`

Examples

```r
fs <- kfamset(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
                 set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))
if(require("Rgraphviz")) {plot(fs)}
```
Reduction of Knowledge Structures

Description

Computes the reduction of knowledge structures.

Usage

```r
## S3 method for class 'kstructure'
reduction(x, operation=c("discrimination", "union", "intersection"), ...)

## S3 method for class 'kfamset'
reduction(x, operation=c("discrimination", "union", "intersection"), ...)
```

Arguments

- `x`: An R object of class `kstructure` or `kfamset`.
- `operation`: The set operation under which the reduction is computed.
- `...`: Other arguments to be passed to methods.

Details

`reduction` performs the reduction of a knowledge structure by computing the minimal subset having the same closure as the knowledge structure. Additionally, it allows for computing the discriminative reduction of a knowledge structure. Such a discriminative reduction is a knowledge structure in which each notion contains a single item.

Value

An R object of the same class as `x` where each subset represents one knowledge state of the resulting reduction.

References


See Also

`kstructure`, `knotions`, `closure`

Examples

```r
kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
                     set("a","b","d","e"), set("a","c","d","e"),
                     set("a","b","c","d","e")))
reduction(kst, operation="discrimination")
```
space_property

Space Property of a Knowledge Structure

Description
Tests for and converts to knowledge space.

Usage

kstructure_is_kspace(x)
kspace(x)

Arguments

x
An R object of class kstructure or (for kspace() only) kbase.

Details
A knowledge structure is considered a knowledge space if it includes one state for the empty set {}, one state for the full set of domain problems Q, and a state for the union of any two knowledge states (i.e., the closure under union).
kstructure_is_kspace takes an arbitrary knowledge structure and tests for its space property.
kspace takes an arbitrary knowledge structure, base, or family of sets and returns the corresponding knowledge space, i.e. its closure under union.

Value
For kstructure_is_kspace a logical value.
For kspace an R object of class kspace where each subset represents one knowledge state of the knowledge space.

References

See Also

kstructure, closure.kstructure

Examples

kst <- kstructure(set(set("a"), set("a","b"), set("a","c"), set("d","e"),
set("a","b","d","e"), set("a","c","d","e"), set("a","b","c","d","e")))

# test for knowledge space
kstructure_is_kspace(kst)

# convert to knowledge space
kspace(kst)
Index

* math
  as.binaryMatrix, 2
  as.famset, 3
  as.relation, 4
  closure, 5
  kassess, 6
  katoms, 7
  kbase, 8
  kdomain, 9
  kfringe, 10
  kneighbourhood, 11
  knneighbourhood, 12
  notions, 13
  kstructure, 14
  kstructure_is_wellgraded, 15
  ktrace, 16
  kvalidate, 17
  lpath, 20
  lpath_is_gradation, 21
  options, 14, 15
  plot, 4, 22
  reduction, 23
  reduction.kstructure, 13
  relation, 4, 14, 15
  set, 3, 7–15
  space_property, 24

as.binaryMatrix, 2, 3
as.famset, 2, 3
as.relation, 4

closure, 5, 5, 23
closure.kstructure, 24

kassess, 6
katoms, 7
kbase, 4, 5, 7, 8, 9, 13, 22, 24
kdomain, 9
kfamset, 4, 5, 9, 22, 23
kfamset(kstructure), 14
kfringe, 10, 11, 12
kfringe_inner(kfringe), 10
kfringe_outer(kfringe), 10
kneighbourhood, 10, 11, 12

knneighbourhood, 12
knotions, 13, 23
kspace, 2, 5, 8
kspace(space_property), 24
kstructure, 2, 4–13, 14, 15–17, 19–24
kstructure_is_kspace(space_property), 24
kstructure_is_wellgraded, 15
ktrace, 16
kvalidate, 17

lpath, 20, 21
lpath_is_gradation, 21

options, 14, 15

plot, 4, 22

reduction, 23
reduction.kstructure, 13
relation, 4, 14, 15

set, 3, 7–15
space_property, 24