CHAPTER 8

Spatial data analysis: overlay operations

In the previous chapter we have seen a number of basic spatial analysis operations used for the retrieval, (re)classification and measurement of point, segment, polygon, and raster maps. All operations described in that chapter dealt with single maps. In this chapter we will look at another set of operations dealing with the combination of several maps. These operations can be grouped together as *overlay operations*.

Overlay operations generally form the core of GIS projects. These operations combine several maps and thus give new information that was not present in the individual maps. New spatial elements are created.

Overlay operations are only performed on raster maps in ILWIS. The raster data structure is particularly suitable for such operations, since all maps used in the analysis have the same georeference. They have the same number of pixels, ordered in lines and columns, the same pixel size and the same coordinates. So when maps are combined, the program can look pixel by pixel to the values in the different maps.

ILWIS has a powerful tool in the combination of maps, called Map Calculation. Many maps can be combined at the same time using arithmetic, relational, or conditional operators and many different functions in formulae typed on the command line of the main window of ILWIS.

Other important tools for the overlay of raster maps are the Cross operation, which calculates the frequency of occurrence of all possible combinations of two maps, and the use of a two-dimensional table, which is a matrix in which the user can define how all classes of two maps should be combined. Before you can start with the exercises, you should start up ILWIS and change to the sub-directory c:\ilwis21\data\usrguide\chap08, where the data files for this chapter are stored.

☞

- Double-click the ILWIS program icon in the ILWIS program group.
- Change the working drive and the working directory until you are in the directory c:\ilwis21\data\usrguide\chap08.

Map Calculation

Map Calculation is an operation with which you can calculate new maps using formulas.

The formulas are typed on the command line of the main window, see Figure 1.1, or using the dialog box of the MapCalc operation. A Map Calculation formula or statement to be executed, consists of an output map name that will contain the result of the calculation, the definition symbol (=), or the assignment symbol (:=), and an expression:

```
Output map = Expression
```

or

```
Output map := Expression
```

The result of a Map Calculation formula is a raster map. This may be a new map, or it may be an existing map, which will be overwritten. There are two ways to generate output maps with Map Calculation:

– When the definition symbol (=) is used, a *dependent map* is created. The data in a dependent map depend on data of the input maps via the formula which was used. When data in the source maps is changed, the results of the calculation in the output map can also be recalculated.

– When the assignment symbol (:=) is used, a map is created that is independent of other maps (a *source data object*, see chapter 2). You can edit the data stored in such a map using the Pixel Editor.

An expression usually contains *operators* and *functions* to specify the calculation to be performed. The map names and the constants that are used in a formula are called *operands*. When the expression is executed, the program will perform the calculation on a pixel by pixel basis, starting from the first pixel in the first line, and continuing till the last pixel in the last line.

The available MapCalc and TabCalc operators and functions are listed in the On-line Help and in the Reference Guide. Table 8.1 presents an overview of the Map Calculation functions.

! You can use the following short-cut keys:
- Ctrl+V(to paste information from the Clipboard to the Command line).
- Ctrl+C (to copy from the Command line to the Clipboard).
This is extremely useful when you have to edit complicated and long formulas. For an overview of the keyboard shortcuts see Appendix I of the Reference Guide.

In the following exercises, you will see a number of examples of Map Calculation formulas, first some that will produce value maps, then some that will give output maps with a class or ID domain. You will also look at how dependency links can be used to update maps made with Map Calculation.

Table 8.1: Some examples of the ILWIS functions used in Map Calculation and Table Calculation. All the functions in this table can be used on maps with a domain type value. The Conditional and Undefined functions can also be used on maps with a domain type class or ID. For a complete overview see the On-Line Help, or the Reference Guide.

Functions	**Syntax**	**Operation**
Conditional IFF	IFF (*a*,*b*,*c*)	If condition *a* is true, then return the outcome of expression *b*, or else return the outcome of expression *c*.
Relational	INRANGE (*a*,*b*,c)	Tests whether values of expression or map *a* are contained by a range or closed interval with endpoints *b* and *c*.
Undefined	ISUNDEF	Tests whether *a* is undefined.

Exponential	SQ(*a*)
	SQ(*a*,*b*)
	SQRT(*a*)
	HYP(*a*,*b*)
	POW(*a*,b)
	EXP(*a*)
		a square; a*a
		a square plus *b* square (a*a + b*b).
		Calculates the positive square root of *a*.
		Calculates the positive square root of the sum of *a* square and *b* square.
		a raised to the power *b*. The n-th root of a is found by: POW(*a*, 1/n).
		Value e (i.e. 2.718) raised to the power *a*.

Logarithmic	LOG(*a*)
	LN(*a*)
		Calculates the 10-based logarithm of *a*.
		Calculates natural logarithm of *a*.

Random	RND(*a*)
	RND(0)
	RND()
		Returns random integer values in the range [1;a]
		Returns a 0 or a 1 at random.
		Returns random real values in the range [0;1>, i.e. including 0, excluding 1].

Sign	ABS(*a*)
	SGN(*a*)
		Returns the absolute (= positive) value of *a*.
		Returns -1 for negative values of *a*, 0 if *a*= 0, & 1 for positive values of *a*.

Rounding	ROUND(*a*)
	FLOOR(*a*)
	CEIL(*a*)
		Rounds *a* to an integer value.
		Returns the largest integer value smaller than input value.
		Rounds up; returns the smallest integer value larger than input value.

MinMax	MIN(*a*,*b*)
	MIN(*a*,*b*,*c*)
	MAX(*a*,*b*)
	MAX(*a*,*b*,*c*)
		Returns the minimum of two expressions *a* and *b*.
		Returns the minimum of three expressions *a* *b* and *c*.
		Returns the maximum of two expressions *a* and *b*.
		Returns the maximum of three expressions *a*, *b* and *c*.

NDVl	NDVI(*a*,*b*)	Calculates the Normalized Difference Vegetation Index of 2 images (*b*-*a*) / (*a*+*b*).

Trigonometric	SIN(*a*)
	COS(*a*)
	TAN(*a*)
	ASIN(*a*)
	ACOS(*a*)
	ATAN(*a*)
	ATAN2(*y*,*x*)
		Sine; returns real values in the range -1 to 1.
		Cosine; returns real values in the range -1 to 1.
		Tangent: sin/cos.
		Arcsin; $\sin^{-1}$ returns real values in radians in the range $-\pi/2$ to $\pi/2$.
		Arccos; $\cos^{-1}$ returns real values in radians in the range 0 to π .
		Arctan; $\tan^{-1}$ returns real values in radians in the range $-\pi/2$ to $\pi/2$.
		Returns the angle in radians of two input values.

Angular conversion	DEGRAD(*a*)
	RADDEG(*a*)
		Degrees to radians function: $a*2\pi/360$.
		Radians to degrees function: $a*360/2\pi$.

Hyperbolic	SINH(*a*)
	COSH(*a*)
	TANH(a)
		Hyperbolic sine: $(e^a - e^{-a})/2$.
		Hyperbolic cosine: $(e^a + e^{-a})/2$.
		Hyperbolic tangent: tanh(*a*) = sinh(*a*)/cosh(*a*).

Pre-defined values and variables	PI EXP(*a*) %X %Y %L %C	Value π : 3.1459265... Returns exponential: e^a .Value e : 2.7182818... Variable to calculate with X-coordinates in a map. Variable to calculate with Y-coordinates in a map. Variable to calculate with Line or Row numbers in a map. Variable to calculate with Column numbers in a map.
Special function to classify values	CLFY(*a* , Domain Group)	Classifies the values of *a* according to a domain Group.

8.1 Map Calculation formulas resulting in value maps

There is a wide range of operators and functions that are used to analyze raster maps with the domain type value. They also work on maps with the domain type image, which is a special type of value domain. In the following sections, firstly some examples of the various operators are shown, before we will apply them in a small case study.

Arithmetic operators

Arithmetic operators are the most simple operators. They are used for multiplication, division, subtraction or addition of maps and/or constant values (see table 8.2). It is obvious that arithmetic operators can only be used on value maps.

Table 8.2: List of the ILWIS arithmetical operators used in the MapCalc with a domain type value or image.

Operators	**Syntax**	**Operation**	**Example**
Arithmetic	+	Add	*a* + *b*
	-	Subtract	*a* - *b*
	*	Multiply	*a* * *b*
	/	Divide	*a* / *b*
	a MOD *b*	Returns the remainder of *a* divided by *b* (returns 1 if *a*=10 & *b*=3)	*a* MOD *b*
	a DIV *b*	Returns the quotient of *a* divided by *b* (returns 3 if *a*=10 & *b*=3)	*a* DIV *b*

In figure 8.1 some examples of these arithmetical operators are given. Let us look at the most simple one:

```
MapC=MapA+10
```

This means: Add a constant factor of 10 to all pixel values of raster map `MapA` and store the result in output map `MapC`. In other words, output `MapC` is equal to the sum of raster map `MapA` and constant 10.

```
MapC=MapA+MapB
```

In words: add the pixel values of `MapA` and `MapB` and store the result in `MapC`.

```
MapC = (MapA - MapB)/(MapA + MapB) * 100
```

This means: Store on disk the raster map `MapC`, resulting from the subtraction of `MapB` from `MapA`, divided by the sum of `MapA` and `MapB` . The result of this is multiplied by 100. This formula when applied on two (2) satellite bands (one with visible or red values and the other near-infra red values) is called the NDVI (*Normalized Difference Vegetation Index*). The output values range from -100 to +100.

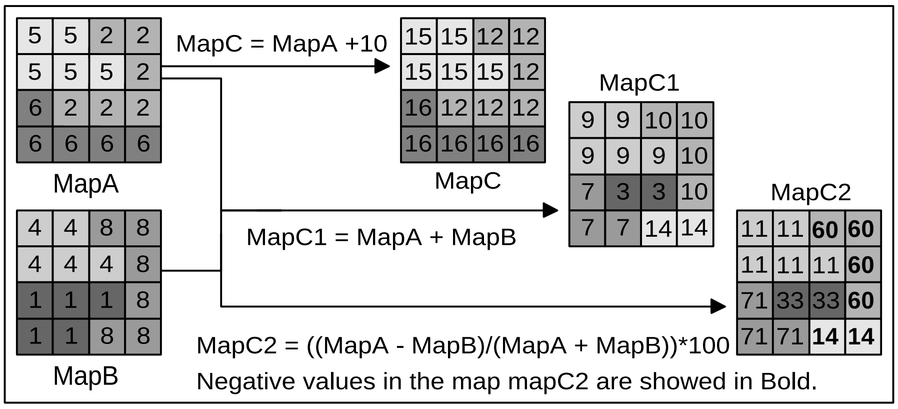


Figure 8.1: Some examples of arithmetic operations in ILWIS. The input maps have domain type value.

Relational operators

Relational operators (see table 8.3) test whether one expression is larger, smaller, equal than another expression, etc.

Table 8.3: Relational operators used in MapCalc for value maps. Only the first and the last one can also be used for class or ID maps.

Operators	**Syntax**	**Operation**	**Example**
Relational	=	Equal to	a = b
	<	Less than	a < b
	<=	Less than or equal to	a <= b
	>	Greater than	a > b
	>=	Greater than or equal to	a >= b
	<>	Not equal to	a <> b

Relational operators are used in combination with logical operators or conditional functions. If we use only a relational operator in a formula, the formula will be a boolean statement. Some examples of boolean statements are:

```
MapC = MapA <> MapB
```

In words: The statement is that `MapA` is different from `MapB`. This statement can be either true or false. The output map, `MapC`, will therefore only contain two different values: `True` or `False` (bool domain). Such a statement would be useful for change detection, for example to compare two land use maps of different periods.

Logical operators

Logical operators (see table 8.4) compare two expressions and check if both are true (AND), at least one is true (OR), only one is true (XOR), or one is not true (NOT).

Table 8.4: Logical operators used in MapCalc. They can be used on maps, with all types of domains

Operators	Syntax	Operation	Example
Logical	AND	Returns true if both expressions *a* and *b* are true.	(*a*) AND (*b*)
	OR	Returns true if one or both of the expressions *a* and *b* is true.	(*a*) OR (*b*)
	XOR	Returns true if only one of the expressions *a* and *b* is true.	(*a*) XOR (*b*)
	NOT	Returns true if expression *b* is false.	NOT (*b*)

These operators are also called boolean operators. An example of a boolean operator (`and, or, xor, not`) is presented in figure 8.2.

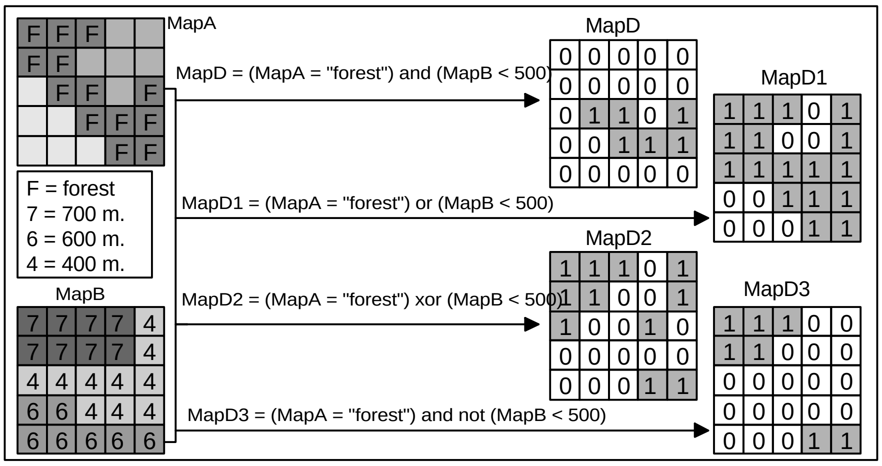


Figure 8.2: Examples of Logical operations in ILWIS. MapA has domain type class and MapB has domain type value.

```
MapD=(MapA=“Forest”)AND(MapB<500)
```

In words: When a pixel in `MapA` has class name `Forest` and at the same time this pixel in `MapB` has a value less than 500, assign value 1 (true) to this pixel in the output map (`MapD`). Assign value 0 (false) to all other pixels.

```
MapD1=(MapA=“Forest”)OR(MapB<500)
```

This means: If a pixel in `MapA` has the class name `Forest` and/or if this pixel in `MapB` has a value less than 500 or the other way around, assign value 1 (true) to this pixel in the output map (returns 1 if one or both conditions are true). Assign value 0 (false) for all pixels where this is not the case.

```
MapD1=(MapA=“Forest”)XOR(MapB<500)
```

In words: If a pixel in `MapA` has the class name `Forest` or if this pixel in `MapB` has a value less than 500, assign 1 (true) to this pixel in the output map (returns 1 if only one of the conditions are true). Assign value 0 (false) for all pixels where this is not the case. This statement is called *exclusive OR*. Only one of the statements can be true, not both.

```
MapD1=(MapA="Forest")AND NOT(MapB<500)
```

In words: when a pixel in `MapA` has the class name `Forest` and at the same time this pixel in `MapB` does not have a value less than 500, assign a 1 to this pixel in the output map (returns 1 if the first condition is true and the second is false). Assign a 0 for all pixels where this is not the case.

Conditional functions

The examples that we have used for the relational and logical operators all give output values which are either true or false. In practice we use these operators mostly with the so-called *conditional iff function*. The general syntax for the conditional iff functions is:

```
Outmap=IFF(Condition,Then statement,Else statement)
```

or

```
Outmap:=IFF(Condition,Then statement,Else statement)
```

in which:

`Outmap` is the name of output map, The symbol `=` is used to create a dependent output map, while the assignment symbol `:=` is used to create an independent (editable) output map. `IFF` is the function, `Condition`refers to the condition to be met, `Then expression` refers to the calculation that has to be performed when the condition is met and the `Else expression` refers to the calculation that has to be performed when the condition is not met. Here are some examples of the use of conditional functions in ILWIS (Figure 8.3):

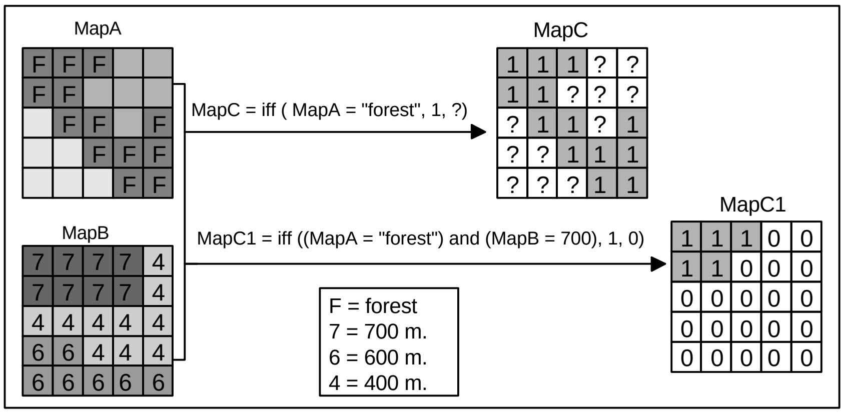


Figure 8.3: Examples of conditional functions in ILWIS. MapA has domain type class and MapB has domain type value.

```
MapC = iff (MapA = "Forest", 1, ?)
```

In words: If a pixel in `MapA` has a class name forest, then assign a value 1 to this pixel in the output map (`MapC`). If the pixel does not have the class forest, then assign the undefined value (?).

```
MapC = iff ((MapA = “Forest”)and(MapB = 700), 1, 0)
```

In words: If a pixel in `MapA` has a class name forest and at the same time this pixel in `MapB` has a value equal to 700, then assign a value 1 to this pixel in the output map; else assigns value 0.

Practicing with operators and functions in a small case study

After this overview of different operators and functions, let us apply these in the analysis of a simple, hypothetical, problem. Suppose we want to calculate the price of the land in the Cochabamba region. The *average land prices* per hectare are given in an attribute table linked to the land use map. However, these average values may be lower, depending on a set of two criteria:

– 1. The price of the land will be 100 percent of the average value when located on slopes of less than 20 degrees, and 70 percent when located on slopes of more than 20 degrees. Slope information is stored in the map `Slope`.

– 2. The price of the land will be 40 percent of the average value when it is located on an active landslide or in an area with high erosion, and 60 percent when located on an old landslide. For this criterion we need the geomorphologic map (`Geom`).

When evaluating the combination of criteria we only look at which of the criteria will lead to the lowest land price. So if a piece of land is located on an active landslide, the land value is only 40 percent of the average price. If the same piece of land is also located on a slope more than 20 degrees, which would lead to a decrease of 70 percent of the average, the value of the land is still 40 percent of the average, since 40 percent of the average is less than 70 percent of the average land price. This is why we will treat the two criteria independently (using the same land price data), and obtain the final result by taking the minimum of the two.

Please keep in mind that the objective of this exercise is not that you learn about an application - for that the problem is too hypothetical - but that you learn to use map calculation formulas. The ILWIS Applications Guide focuses more on applications than on tools.

Before we start with the analysis using Map Calculation formula, let us first have a look at the input data with the pixel information window.

☞

- Open the map `Landuse`.
- Open the pixel information window and add the raster maps `Landuse, Slope,` and `Geom.`

- Move with the mouse pointer through the `Landuse` map and consult the information of the three maps in the pixel information window.
- Close the map window and the pixel information window when you think you have a good idea of the content of the maps and attribute tables connected to it.

You start the analysis with an operation that you are already familiar with (chapter 6): *reclassification*. The land use map has an attribute table, in which the average land value (per hectare) is stored for each land use type. So you will *reclassify* the class map `Landuse` with the `Landvalue` column, which will result in a value map. Since the average land values are given per hectare (10000 m^2), and you are working on maps with a pixel size of 20 meters (i.e., 400 m^2 per pixel), you need to divide the land values by 25 in order to obtain the average value per pixel.

☞

- Type the following formula on the command line of the main window:
  `Landval=(Landuse.Landvalue)/25`
- Press Enter.

In this formula, you combine a reclassification expression (of the form: map.column) with an arithmetic operator (divide). Unlike in earlier ILWIS DOS versions, the table name should not be mentioned unless the table is not linked to the map in the Properties dialog box.

☞

- Accept the defaults in the Raster Map Definition dialog box and click OK.
- Display map `Landval` with representation `pseudo` and close it when you have seen the result.

Now you will take into account the first criterion: If the slope is more than 20 degrees, the land price will only be 70 percent of the average.

☞

- Type the following formula:
  ```
  Landval1 = iff(Slope > 20 , Landval * 0.7 ,
  Landval)
  ```

- Press Enter.

In this formula, you combine an IFF function with a relational operator in the 'conditional part', and an arithmetic operator in the 'then part'. Pixels in the new `Landval`, or they map `Landval1` either have a value which is only 70 % of those in the map have the same value as the original `Landval` map.

☞

- Accept the defaults in the Raster Map Definition dialog box and click OK.
- Display the map `Landval1` with representation `pseudo` and close it when you have seen the result.

Now let's look at the second criterion: If a pixel is located on an old landslide, then the value is only 60 percent of the average land value. If the pixel is on an active landslide or on an active erosion area, the value is only 40 percent of the average.

☞

- Type the following formula:
```
Landval2 = iff(Geom="ol", Landval * 0.6, iff((Geom
= "al") or (Geom = "he"), Landval * 0.4 , Landval))
```
- Press Enter.
- Accept the default in the Raster Map Definition dialog box.
- Click OK.

There are several things that need to be explained about this formula. First of all, we have an example here of a *nested IFF function*, i.e. an IFF function within another one. When you use a nested IFF function, you have to make sure that every individual IFF function has the syntax IFF (condition, then expression, else expression); i.e. an opening bracket, three components separated by commas and a closing bracket. Further more, in this formula, codes are used instead of the names of the geomorphologic class names. The unit "`old Landslide`" in the domain `Geom` has the code "`ol`". If you use codes, the formulas can be much shorter.

Lastly, the relational operator OR was used, since both the geomorphologic units `"active Landslide"`, code "`al`" and `"heavy erosion"`, code `"he"` will give the same lowering of the land values.

Now you have generated two maps each based on one criterion (`Landval1`, and `Landval2`). What should you do for pixels where more than one of these criteria occur, e.g.: pixels with a slope more than 20 degrees, located on an active landslide? As explained before, the minimum condition determines the result (since

40% of a value is less than 70% of the same value). Therefore, the minimum of the two maps is taken. For this you can use the function: min.

☞

- Type the following formula:
  `Landvalc=min(Landval1,Landval3)`
- Press Enter.

If you make typing errors in a formula, the program will give an error message in which the location of the error is indicated with a ^ symbol.

☞

- Click OK. Make sure that the command line is active and press the up arrow, so that the formula reappears. Correct the formula so that it shows:
  `Landvalc=min(Landval1,Landval2)`
- Press Enter. Click OK in the Raster Map Definition dialog box.
- Display the map with representation `pseudo`.
- Display the map `Landval` with same representation and position the map window next to that of map `Landvalc`. Use the mouse pointer to find out the values in both maps.

Undefined values

Note that the colored areas in the map `Landvalc` are occupying a smaller part of the map than in the map `Landval`. The white areas in both maps represent undefined values. Since the geomorphologic map has a larger part which was not mapped (so a larger part with undefined values), the formulas, in which this map was used, resulted in maps which have undefined values for those pixels where any one of the input maps has undefined values. An undefined value in ILWIS can mean several things:

– No data is available for the pixel. In this case a question mark in a map indicates that the part of the area lacks data, and data should be supplied in order to do the analysis properly.

– The pixel is located outside the study area. Pixels with question marks indicate that the pixel is outside the study area. No action is required from the user.

– The result of a calculation was wrong. In this case a question mark indicates that a certain operation was wrongly made. This may happen when you make a typing error in the names of classes in a formula. For example if you write the following formula:

`Result=iff(Landuse="Forrest",Landuse,"?")`

This formula results in a map with undefined values, since the correct class name is `Forest`. This can be corrected by changing the definition of the output map, and recalculating it.

– The values in the output map, with a value domain, fall outside of the value range defined in the domain. For example, when you write the following formula:

  `result = Dem * 10` , and select the value range 0 to 1000 for the output map.

  This formula results in a map with undefined values, since the value range of `Dem` was 2500 to 4600, and the expected output values should have the value range 25000 to 46000. This can be corrected by increasing the value range for the output map, and recalculating the map.

So when you obtain undefined values in an output map, you should check one of these four possibilities.

Special emphasis should be paid to the use of undefined values in IF functions. If you use an IF function that has the form `IFF(a,b,c),` you can have the following possibilities:

– Statement *a* is true, so the result will be *b*,

– Statement *a* is false, so the result will be *c*, or

– Statement *a* is *undefined*. If we don't know what *a* is, then we also can not say whether *a* is true or false, so the result is also undefined.

Note that the situation is more complex, when the condition consists of several statements, combined with logical operators. The result of combining two statements with an AND and an OR statement is shown in the *truth tables* below.

a AND b	**a=True**	**a=False**	**a=Undefined**
b=True	True	False	Undefined
b=False	False	False	False
b=Undefined	Undefined	False	Undefined

a OR b	**a=True**	**a=False**	**a=Undefined**
b=True	True	True	True
b=False	True	False	Undefined
b=Undefined	True	Undefined	Undefined

So for example, if we have the following IF Function: `IFF( a AND b, c, d):`

– If *a* is true and *b* is true, then the condition is true, so the result is *c*.

– If a is true and *b* is undefined, then we cannot know whether the result is true or false, because if *b* were true the result would be *c*, else *d*. Therefore, the result is undefined.

– If *a* is false and *b* is undefined, then already one of the parts of the condition is false, so it doesn't matter anymore what *b* is, because the condition is false anyway, and the result will be *d*.

A different situation occurs, if we have the IF Function: `IFF( a OR b, c, d)`

– If either *a* or *b* is true then it doesn't matter what the other part is (false or undefined), because the condition is true, and the result is *c*.

– If *a* is false and *b* is undefined, then it may be that *b* is true, so the result is undefined.

In most cases this is a logical assumption: You don't know the result of a formula if one of the operands is undefined. However, in this situation, we may still want to use the original land values, linked to the `Landuse` map, even if we don't have any information on the geomorphology.

☞

- Open the map `Geom`, and display the maps `Geom`, `Landval`, and `Landvalc` next to each other. As you can see the map `Geom` contains information for a smaller area than the map `Landval`. Close the three map windows.
- Type the following formula:
  `Landvalf=iff(isundef(Geom),Landval1,Landvalc)`
- Press Enter.

The function `isundef(Geom)` tests whether the map `Geom` is undefined.

In words this formula means: If a pixel in the map `Geom` is undefined (which means no information on geomorphology is available), than we take the value from the map `Landval1` (which includes the slope related land prices), otherwise the value from the map `Landvalc`.

☞

- Accept the defaults in the Raster Map Definition dialog box, and click OK.
- Display the map `Landvalf` with representation `pseudo`. Compare it with the other maps, and close all map windows.

Summary: Map Calculation formulas resulting in value maps

- Map Calculation is an operation with which you can calculate new maps using formulas.
- The formulas are typed on the command line of the main window.
- Map Calculation formulas use raster maps as input and produce a raster map as output.
- There is a wide range of operators that can be used:

  Arithmetic operators, to multiply, divide, subtract or add maps and/or constant values.

  Logical operators, to compare two expressions and check if both are true (AND), at least one is true (OR), only one is true (XOR). The NOT statement checks whether an expression is not true.

  Relational operators to test whether one expression is larger, smaller, equal than another expression etc.
- The most important function is the IFF function, which has the following structure: IFF (expression, then statement, else statement).
- IFF functions can be nested, i.e. an IFF function can occur within another one.
- A map calculation formula can also be used to reclassify a map according to data in an attribute table.
- Formulas result in maps which have undefined values for those pixels where any one of the input maps has undefined values. You don’t know the result of a formula if one of the operands is undefined, depending on the operators that are used.
- The function isundef(*a*) tests whether expression *a* is undefined. With this function it is possible to change undefined values into known values.

8.2 Map Calculation and dependencies

In the previous exercise we have made a few maps, based on the input maps `Geom`, `Slope`, and `Landuse`. We also used the attribute table `Landuse` in which the average value of the land per hectare was indicated.

On the basis of these source maps (in fact `Slope` is not a real source map since it is made from a digital elevation model) four *dependent maps* were made: `Landval`, `Landval1`, `Landval2`, `Landvalc`, and `Landvalf`.

Now suppose that the average value of the land has changed, for some economic reason. This would mean that we have to recalculate all the result maps again.

This is where the concept of *dependency* becomes very useful. In chapter 2, we have seen the basic concept of dependency. Dependent maps know how they are made, and whether they are up-to-date. As soon as one of the input maps is changed, the dependent map knows that it is no longer up-to-date.

☞

- Open the Properties dialog box of the map `Landvalf`.

The last line above the buttons indicates: Object is up-to-date. This means that no changes were made in the input maps after the time that the map `Landvalf` was created.

☞

- Close the Properties dialog box of the map `Landvalf.`
- Open the table `Landuse`.

This table contains the column `Landvalue` (the average value of the land per hectare for the different land use types). The values for the units `Lake` and `Riverbed` are undefined.

Now suppose there is a shortage of water in the Cochabamba area, then the price of water will increase considerably. Many factories would like to have their own lake, as is the case for the large beer brewery (Taquina). So there should also be a value added to the land use type : `Lake`.

☞

- Edit the field in the column `Landvalue` for the land use type `Lake` and type the value : `400`
- Close the table `Landuse`. Open the map `Landvalf`.
- Open the pixel information window and add the raster map

Landuse to it. Find out where the lakes are.

As you can see the lakes in the map Landvalf still have undefined values. Although we have edited the land values in the table Landuse for the lakes, the final result is still not updated. Updating does not happen automatically, but is decided by the user.

☞

- Close the map Landvalf.
- Close the pixel information window.
- Open the Properties dialog box of the raster map Landvalf again.

The last line in the Properties dialog box of the raster map Landvalf now reads: `Object is not up-to-date: Column Landuse.Landvalue (Day month, time, year).`

It is indicated (day, month, time, year) when you have updated the column Landvalue (if it is correct that should be a minute ago).

☞

- Press the button: Make Up to Date.
- Click Yes to answer the question: `Map Calculate “Landvalf” is not up-to-date. Recalculate it to make it up-to-date?`.

The program now starts to recalculate all the maps that were used to make the final map Landvalf. In fact it will do the previous exercise for you again. First the map Landval is recalculated, in which the column Landvalue from the table Landuse was used (the one that you just updated). Then the maps Landval1, and Landval2 are recalculated. These are combined in the map Landvalc, after which the undefined values are removed and the final map Landvalf is made. The entire recalculation will take a few minutes. You will see the tranquilizers of the various calculations. When the tranquilizer disappears the calculation is finished.

☞

- Open the raster map Landvalf.
- Click the location of one of the lakes.

As you can see the map `Landvalf` is now updated.

☞

- Close the raster map `Landvalf`.

Summary: Map Calculation and dependency links

– When the definition symbol (=) is used in a Map Calculation formula a *dependent map* is created. Data in a dependent map depends on data of input maps via the formula which was used.

– When data in the source maps is changed, this is indicated in the Properties dialog box of the dependent map. The map is then no longer Up-to-date.

– When the button Make up-to-Date is pressed, the dependent map will be recalculated.

– When a map is no longer up-to-date, and the input maps are also no longer up-to-date, they are also recalculated, in a chain of recalculations, up to the source data that were originally changed.

– The dependency of maps allows for easy updating.

8.3 Map Calculation formulas resulting in class or ID maps

When we work with IF functions that give class or ID results, we may have four different situations:

– 1. *IFF(expression, domain1,"?").* The result of an expression may be an existing class or ID domain (mostly of one of the input maps), or it may be an undefined value.

– 2. *IFF(expression, domain1,"name").* The result of an expression may be an existing domain or a name which is not in the domain. In that case you cannot simply use the existing domain (domain1) as the domain of the output map. Since the new name is not in the domain, the map will contain undefined values for those pixels. You either have to add the "name" to the domain or create a new domain. A similar situation is *IFF(expression,"name",domain1).*

– 3. *IFF(expression, domain1, domain2).* The result of an expression may be a combination of two existing domains (domain1 and domain2).

– 4. *IFF(expression, "name","name").* The result of an expression may be two names which are not in an existing domain.

In situations 2, 3, and 4 you cannot simply use an existing domain for the output maps. You either should add items to an existing domain, or you should create a new domain.

When ILWIS encounters in an IF function one of the last three possibilities, the program will suggest that you use an existing domain, unless two new names are in the IF function. When you decide to use the default existing domain and press OK in the Column properties dialog box, you will get a warning, and you are asked whether you want to add the missing items to that domain. However, it is not advisable to generate large domains with a mixture of information. In many cases it is better to generate a new domain. To do so, press the create domain button in the Raster Map Definition dialog box . You will now look at the 4 situations with some examples.

IFF(expression, domain1,"?")

If the operation uses only one map, this is in fact a retrieval expression, as we have seen in chapter 7. Note that you have to use a question mark between double quotes to assign undefined values in a map with a class or id domain. For example: To find the areas with landslides you can type the following formula:

☞

- Type the following formula on the command line:
  ```
  Slide = iff((Geom = "al") or (Geom = "ol"), Geom,
  "?")
  ```
- Press Enter.

Note that in the Raster Map Definition dialog box, the default domain for the output map is `Geom`. This is ok.

☞

- Accept the defaults by clicking the OK button in the Raster Map Definition dialog box.
- Display the map `Slide`, check its contents and close it.

In an expression, you can use different maps with different domains. In the next example, if you want to find out the geological unit of the landslides that are between 3500 and 4000 meters:

☞

- Type the following formula on the command line:
```
Slide1=iff(((Geom=“al”)or(Geom=“ol”))and
inrange(Dem,3500,4000),Geology,“?”)
```
- Press Enter. Click OK in the Raster Map Definition dialog box.

In the expression part of this formula, we first evaluate whether the geomorphologic unit is either an active or an old landslide, and secondly whether the altitude is between 3500 and 4000 meters.

The special function in the expression `inrange(Dem,3500,4000)` checks whether the values in the map `Dem` are between 3500 and 4000.

Furthermore, the part dealing with the geomorphologic units is put between brackets, because otherwise the statement would be very different: If a pixel in the geomorphologic class is “`active Landslide`” or, on the other hand, if it is an “`old Landslide`” located between 3500 and 4000 meters.

☞

- Display the map `Slide1`, check its contents and close it.

IFF(expression, domain1,”name”)

In this example, we explain an IF function in which the “then” part uses an existing domain and the “else” part contains an item which is not in the domain. You can add the new “name” to the existing domain or create a new domain.

Suppose we want to find out where the landslides are. If there is no landslide, we will use the new word “`no Landslide`”.

☞

- Type the following formula on the command line:
  `Slide2=iff((Geom="al")or(Geom="ol"),Geom,"no Landslide")`
- Press Enter. The Raster Map Definition dialog box indicates the default domain `Geom`.
- Click OK .

Now the Merge Domain dialog box appears with the question:
`Add string "no Landslide" to the domain "Geom".`

If you answer with `No`, the name "`no Landslide`" will not be added as a class to the domain `Geom`. Since the name "`no Landslide`" is not in the `Geom` domain, the program will treat the pixels that should obtain the name "`no Landslide`" as undefined pixels, and the map will be exactly the same as the one you previously made.

If you answer `Yes`, the name "`No landsilde`" is added to domain `Geom`.

☞

- Answer `Yes` to the question: `Add string "no Landslide" to the domain "Geom".`
- Display the map `Slide2` and look at the result.
- Close the map window .

A formula may also have the form *iff(expression,"name",domain1)*. For example, if we want to make a map where "`old Landslide`" and "`active Landslide`" are changed to "`Landslide`", and for the rest of the map the geomorphologic units are shown.

☞

- Type the following formula on the command line:
  `Slide3=iff((Geom="al")or(Geom="ol"),"Landslide", Geom)`
- Press Enter. The Raster Map Definition dialog box indicates the default domain `Geom`.
- Click OK.
- Answer the question: `Add string "Landslide" to the domain "Geom"` with Yes. Display the map `Slide3` and close it after checking.

IFF(expression, domain1, domain2)

The result of an expression may be a combination of two existing domains (domain1 and domain2).

For example: we want to add the items "`old Landslide`", "`active Landslide`" and "`heavily eroded area`" to the map `Landuse`.

☞

- Type the following formula on the command line:
  `Slide4=iff((Geom="al")or(Geom="ol")or(Geom="he"), Geom,Landuse)` ↵
- The Raster Map Definition dialog box indicates the default domain `Landuse`.
- Click the Create button. The Create Domain dialog box is opened.
- Type for the domain name: `Slide4`. Click OK.
- The Domain Editor is now opened. We will not add any items to the domain. This will be done automatically while performing the calculation.
- Close the Domain Editor. Now you are back in the Raster Map Definition dialog box. You will see that the domain is now `Slide5`.
- Click OK.
- Answer the question: `Merge strings of domain "Geom" into domain "Slide4"` with `Yes`.
- Answer the question: `Merge strings of domain "Landuse" into domain "Slide4"` with `Yes`.
- Open the map `Slide4`.
- Open the domain `Slide4`.
- Close the map and the domain after checking their contents.

Now the two domains `Geom` and `Landuse` are merged into a new domain: `Slide4`. This is much better than adding the contents of one domain to the other, since the domains of the original maps remain unchanged.

IFF(expression, "name","name")

In this last example, we will look at a formula where both the THEN and ELSE parts contain new names. In this situation, ILWIS does not know the output domain, and will not provide you with a default domain in the Raster Map Definition dialog box.

To continue with our example on landslides: You will now make a map with only two units: "`Landslides`" or "`no Landslides`". In this case, when the result of a formula can have two possibilities, it is best to create a new domain first in which you enter the two names.

☞

- From the File menu of the Main window, select Create and Create Domain. The Create Domain dialog box is opened.
- Type `Slide5` for the Domain name. Make sure the option Class is selected and click OK. The Domain Editor is opened.
- Press INS and add the class "`Landslides`". Click OK. Press INS again and add the class "`no Landslides`". Close the Domain Editor.
- Type the following formula on the command line:
  ```
  Slide5=iff((Geom="al")or(Geom="ol"),"Landslides",
  "No landslides")
  ```
- Press Enter. The Raster Map Definition dialog does not indicate a default domain.
- Select the domain `Slide5`. Type the description: `Presence of Landslides`. Click OK.
- Open the map `Slide5`.
- Close the map and the domain after checking their contents.

Note that you can also use a Bool domain (containing the possibilities "`true`", "`false`" and undefined) or a yesno domain for the output map. This domain uses "`yes`" and "`no`" instead of "`true`" and "`false`". However the yesno domain cannot be used in an IF function. In that case the formula would be:

```
Slide6=(Geom="al")or(Geom="ol")
```

Select the domain yesno in the Raster Map Definition dialog box.

Summary: Map Calculation formulas resulting in Class or ID maps

– When the result of an IFF statement is not a value, you can have 4 possibilities:

  1. IFF(expression, domain1,”?”).

  2. IFF(expression, domain1,”name”).

  3. IFF(expression, domain1, domain2).

  4. IFF(expression, “name”,”name”).

– For each of these combinations, you can choose to add classes or IDs to an existing domain, to merge two domains, or to create a new domain.

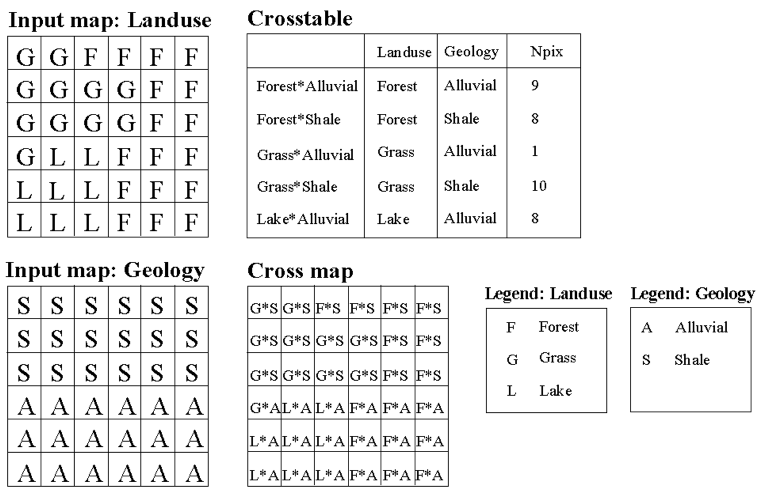
– When two domains contain related information, one domain could be merged into the other domain. When two domains contain information of different kinds, it is better to generate a new domain.

8.4 The Cross operation

When we are interested in many combinations of two maps, Map Calculation formulas as discussed in the previous exercises, are not useful anymore. For example, to combine two maps, `MapA` with 5 classes, and `MapB` with 3 classes, we may need about 15 nested IFF functions within the same formula; this would obviously become too complicated. Often, maps have even more classes. We use another operation: Cross.

The Cross operation performs an overlay of two raster maps by comparing pixels at the same positions in both maps and keeping track of all the combinations that occur between the values or classes in both maps. The input maps used in a Cross operation should be raster maps that have the same georeference. During the Cross operation, combinations of class names, identifiers or values of pixels in both maps is listed, the number of pixels occurring as this combination is counted, and the areas of the combinations are calculated. The results are stored in an *output cross map* and a *cross table*. The output cross table obtains an Identifier domain, with the same name as the output map. This domain contains items which are combinations of the class names, IDs, group names or values of the first input map and those of the second input map.

A simple example of a Cross operation between two class maps is shown in Figure 8.4.



Input map: Landuse

G	G	F	F	F	F
G	G	G	G	F	F
G	G	G	G	F	F
G	L	L	F	F	F
L	L	L	F	F	F
L	L	L	F	F	F

Crosstable

	Landuse	Geology	Npix
Forest*Alluvial	Forest	Alluvial	9
Forest*Shale	Forest	Shale	8
Grass*Alluvial	Grass	Alluvial	1
Grass*Shale	Grass	Shale	10
Lake*Alluvial	Lake	Alluvial	8

Input map: Geology

S	S	S	S	S	S
S	S	S	S	S	S
S	S	S	S	S	S
A	A	A	A	A	A
A	A	A	A	A	A
A	A	A	A	A	A

Cross map

G*S	G*S	F*S	F*S	F*S	F*S
G*S	G*S	G*S	G*S	F*S	F*S
G*S	G*S	G*S	G*S	F*S	F*S
G*A	L*A	L*A	F*A	F*A	F*A
L*A	L*A	L*A	F*A	F*A	F*A
L*A	L*A	L*A	F*A	F*A	F*A

Legend: Landuse

F	Forest
G	Grass
L	Lake

Legend: Geology

A	Alluvial
S	Shale

Figure 8.4: Example of a cross operation. The maps Landuse and Geology are combined. A cross table and a cross map are made.

In the following exercise you will do two Cross operations:

– Between two class maps: `Landuse` and a classified slope map, in order to find the percentages of flat, moderate and steep slopes for each land use type.

– Between an identifier map (`Catchmen`) and a value map (`Drainage`) in order to calculate the drainage density per catchment.

Crossing two class maps

Suppose you want to know the percentage of each land use type located on flat, moderately steep, and steep slopes. In order to calculate that, we need two input maps: The raster map `Landuse`, and a classified slope map (`Slopecl`). This map has three classes: `flat` (0-10 degrees), `moderately steep` (10-25 degrees), and `steep` (> 25 degrees). Note that some examples of classifying value maps were already presented in exercise 6.5.

☞

- Display the maps `Landuse` and `Slopecl` and check the meaning of the units. Close the map windows.
- Double-click the operation Cross in the Operations - list. The Cross dialog box is opened.
- Select raster map `Landuse` in the list box 1st Map.
- Select raster map `Slopecl` in the list box 2nd Map.
- Type `Luseslop` in the text box Output Table.
- Click the check box Show. Type for the description: `Combination of Landuse and Slopecl.`
- Click the check box Output Map.
- Type `Luseslop` in the text box Output Map.
- Click OK. The output cross table is displayed.

It contains quantitative information on the combinations of the two input maps.

☞

- Display the output cross map `Luseslop` and check the meaning of the units by clicking them. As you can see the name consists of the names of the maps `Landuse` and `Slopecl`.
- Close the map `Luseslop`. Activate the table window.

In the table all combinations of the land use and slope classes are shown, together with the number of pixels and the area. Now you will use this cross-table to calculate the percentages of the three slope classes occurring within each land use type. Firstly you will calculate three columns, `flat`, `moderate` and `steep`, in which only the records that are actually a combination with flat, moderate or steep slopes will have a value for the area, and not the others.

Then you apply Aggregation functions combined with table joining as you have seen in chapter 5. The aggregation function sums the values of column `Area`, grouped by the `Landuse` classes, and stores the result in the column `Totarea` of table `Landuse`.

☞

- Type the following formula on the command line of the table window:

  `Flat= iff(Slopecl=“flat”,Area,0) ↵`
- Click OK in the Column Definition dialog box.
- Press the up-arrow and edit the previous formula, so that it is:

  `Moderate = iff(Slopecl=“moderately steep”,Area,0) ↵`
- Click OK in the Column Definition dialog box.
- Press the up-arrow and edit the previous formula, so that it is:

  `Steep = iff(Slopecl=“steep”,Area,0) ↵`
- Click OK in the Column Definition dialog box.
- From the Columns menu in the table window select Aggregation. The Aggregate Column dialog box is opened.
- Select the Column: `Area`, select the Function: `Sum`, select the check box Group by and select the column `Landuse`, select the check box Output table, and type the table name: `Landuse`. Type for the output column: `Totarea`.
- Click OK in the Aggregate Column dialog box.
- From the Columns menu in the table window select Aggregation. The Aggregate Column dialog box is opened.
- Select the Column: `Flat`, select the Function: `Sum`, select the check box Group by and select the column `Landuse`, select the check box Output table, and type the table name: `Landuse`. Type for the output column: `Flat`.
- Click OK in the Aggregate Column dialog box.
- Repeat this for the columns `Moderate` and `Steep`, so that the `Landuse` table will contain the columns `Totarea`, `Flat`, `Moderate` and `Steep`.

☞

- Close the table Luseslop.
- Open the table Landuse. As you can see it contains the columns Totarea, Flat, Moderate and Steep that you just generated.
- Type the following formula on the command line of the table window:

  `Pflat= 100*(flat/Totarea) ↵`
- Select the domain perc (percent) in the Column Properties dialog box, and enter for the Precision 1.0. Click OK.
- Create also the columns Pmod and Psteep with similar formulas. Select domain perc for these columns.

The columns Pflat, Pmod and Psteep contain the percentages of flat, moderately steep and steep areas for each land use type.

The Cross operation and the subsequent aggregations and table joining, were made to obtain the percentage of flat, moderately steep and steep land, for each land use type.

We only used the cross table in the exercise. The cross map is often less important because it usually contains far too many combinations. In other types of analysis, for example when you want to make a unique combination map of several input maps, you will also need the cross map as well.

Crossing an ID and a value map: Drainage density.

The Cross operation is not restricted to class maps. It is also possible to use ID or value maps. You should be careful with value maps though, since these may have an enormous number of possible values, and the cross-tables may become too large to handle.

We will work now on the crossing of an identifier map: Catchmen (which contains the catchment areas in the mountainous part of the area) and a value map (Drainage). The map Drainage was created in the previous chapter, in the exercise on drainage density (exercise 7.7). The drainage map contains the length of drainages within each pixel.

☞

- Open the Operations menu in the main window, select Raster Operations, and choose the Cross command.
- Select raster map Catchmen in the list box 1st Map.
- Select raster map Drainage in the list box 2nd Map.
- Type Catchdr in the text box Output Table.

- Click the check box Show. Type for the description: `Cross of catchmen and drainage length per pixel`. Click OK.

The result is a large cross-table which contains the combinations of the catchment names and the number of pixels with a certain drainage length per pixel. The total area of each catchment, as well as the total drainage length per catchment is needed. The first thing to do is to calculate for each catchment the total length. The map drainage shows the length for each pixel. So if you multiply this value by the number of pixels, you know for each length interval what the total length is.

☞

- Type the following formula on the command line of the table window, containing the table `Catchdr`:

  `Length = Drainage * npix ↵`
- From the Columns menu in the table window select Aggregation. The Aggregate Column dialog box is opened.
- Select the Column: `Length`, select the Function: `Sum`, select the check box Group by and select the column `Catchmen`, select the check box Output table, and type the table name: `Catchmen`. Type for the output column: `Totlength`.
- Click OK in the Aggregate Column dialog box.

With this aggregation formula, the total length of the drainage lines within each catchment is calculated and stored in the table catchmen. Now you also need to know the total area of each catchment. The output column `Totarea` is calculated by using the sum aggregation function on the column `Area`, grouped by the catchments (column `Catchmen`).

☞

- From the Columns menu in the table window select Aggregation. The Aggregate Column dialog box is opened.
- Select the Column: `Area`, select the Function: `Sum`, select the check box Group by and select the column `Catchmen`, select the check box Output table, and type the table name: `Catchmen`. Type for the output column: `Totarea`.
- Click OK in the Aggregate Column dialog box.
- Close the table `Catchdr`.

Now that you know the total length of drainage lines in each catchment and the total area of each catchment, you can calculate the drainage density for each catchment.

☞

- Open the table `Catchmen`.
- Type the following formula on the command line:
  `Draindens=1000*Totlength/Totarea`
- Press Enter and click OK in the Column properties dialog box.

The drainage density is expressed in kilometers per square kilometers. Since the original data is in meters and square meters, we need to multiply with 1000. Now you see the final result: the drainage density per catchment.

☞

- Close the table `Catchmen`.

Summary: Cross operation

– The Cross operation performs an overlay of two raster maps by comparing pixels at the same positions in both maps and keeping track of all the combinations of values or classes that occur in both maps.

– The input maps of a Cross operation are raster maps which have the same georeference.

– The output Cross table lists the combinations of class names, identifiers or values of the pixels in both maps. The number of pixels occurring for each combination of classes/ID's or values, from both maps, is counted. The area is obtained by multiplying the number of pixels with the pixel area (square of the pixel size).

– The results are stored in an output cross-map and a cross-table. These obtain an Identifier domain. This domain contains items which are combinations of the class names, IDs, group names or values, of the first input map and those of the second input map.

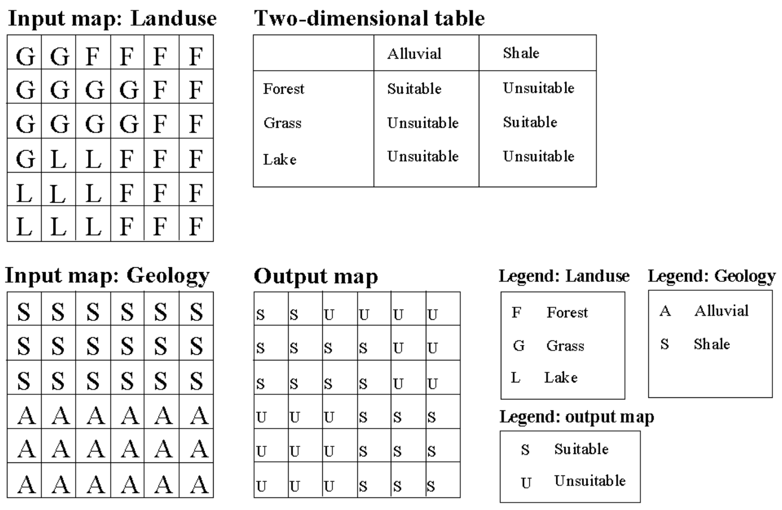
– A Cross-table is often used in combination with Aggregation functions in the table window.

8.5 Two-dimensional tables

As could be seen in the previous exercise, crossing two maps results in a cross-table and a cross-map, in which all possible combinations of the units from the two input maps, are stored as unique identifiers.

In some cases it is better to select the output results yourself, so that you can decide for each combination of classes or IDs in the input maps, what will be the resulting class, ID or value in the output map. This is possible using a so-called *two-dimensional table*. A two-dimensional table is a matrix where each row/record represents a class, or an ID in the first input map, and each column represents a class, or an ID in the second input map.

Each field in the table (each intersection of a record and a column) represents a specific combination of two classes and or IDs in the two input maps (see figure 8.5). The domain type for the input maps should be either class or ID. The domain type of the field in the two-dimensional table, and of the output map, can be class, ID or value. In this example (see figure 8.5), the fields in the two-dimensional table use a class domain with two classes `suitable`, and `unsuitable`. The records and columns in the table are defined by two class domains (`Geology` and `Landuse`).



Input map: Landuse

G	G	F	F	F	F
G	G	G	G	F	F
G	G	G	G	F	F
G	L	L	F	F	F
L	L	L	F	F	F
L	L	L	F	F	F

Two-dimensional table

	Alluvial	Shale
Forest	Suitable	Unsuitable
Grass	Unsuitable	Suitable
Lake	Unsuitable	Unsuitable

Input map: Geology

S	S	S	S	S	S
S	S	S	S	S	S
S	S	S	S	S	S
A	A	A	A	A	A
A	A	A	A	A	A
A	A	A	A	A	A

Output map

S	S	U	U	U	U
S	S	S	S	U	U
S	S	S	S	U	U
U	U	U	S	S	S
U	U	U	S	S	S
U	U	U	S	S	S

Legend: Landuse

F	Forest
G	Grass
L	Lake

Legend: Geology

A	Alluvial
S	Shale

Legend: output map

S	Suitable
U	Unsuitable

Figure 8.5: Example of the use of a two dimensional table, used to combine two input maps. The user can determine for each combination of classes in the input maps, what will be the result in the output map. Grass and Shale, and Forest and Alluvial result in Suitable.

The number of combinations in two maps should be fairly limited, since you have to manually enter the resulting class, ID or value for each specific combination of input classes and/or Ids.

Therefore, the selection of a value map as input for a two-dimensional table is not possible: It will result in too many combinations. Also, for large ID maps this method will not be very suitable. Take for example the city block map, with 717 identifiers. If we would use it in combination with the land use map (which has 12 classes) you would have to fill in 717*12 = 8604 different combinations. This method is thus mostly used for combining class maps with relatively few classes.

In this exercise, we will evaluate which areas are suitable for the extraction of gravel material, which is used for building purposes (road-construction or in the production of concrete for buildings). We do this by combining two maps: `Geology` (containing information on the geological units) and `Landuse` (with the land use types). For each combination of a geological and a land use unit, we will indicate whether it is suitable or unsuitable for gravel extraction.

☞

- In the Catalog, click with the right mouse button on domain `Geology`. Select the command Create 2-dimensional table from the context sensitive menu. The Create 2-dimensional table dialog box is opened.
- Enter for the Table Name: `Gravel`. Type for the description: `Suitability for gravel extraction`. Select the Secondary domain: `Landuse`. The Primary domain: `Geology` is already indicated.
- Click the Create domain button next to the list box of the domain. The Create domain dialog box is opened.
- Type for the Domain Name: `Gravel`. Type the description: `Suitability for gravel extraction`. Click OK. The Domain Editor is opened.
- Press for the INS key, and enter for the Name: `Suitable`, and the code: S. Click OK.
- Press the INS key, and enter for the Name: `Unsuitable`, and the code: U. Click OK.
- Close the Domain Editor. You are now back in the Create 2-dimensional table dialog box. Click OK.

Now the two-dimensional table `Gravel` is opened. The records show the units of the geological map, and the columns of the units of the land use map.

The suitability classes for each combination of the geological and the land use units are shown in table 8.5. Only the geological units “`glacial deposits`”, “`old alluvial deposits`” and “`recent alluvial deposits`” can be used for the extraction of gravel material, but only in combination with the land use units “`bare soil`”, “`grassland`”, “`riverbed`” and “`shrubs`”.

Table 8.5: Two-dimensional table showing the suitability for gravel extraction for the combination of geological and land use units (U = unsuitable, S= suitable).

Operators	Agri	Agri (irr)	Airp	Bare rock	Bare soil	Fores	Grassland	Lake	River	Shrub	Urb cen	Urb per
Glacial	U	U	U	U	S	U	S	U	S	S	U	U
Lake deposits	U	U	U	U	U	U	U	U	U	U	U	U
Old alluvial dep.	U	U	U	U	S	U	S	U	S	S	U	U
Quartzites	U	U	U	U	U	U	U	U	U	U	U	U
Recent alluvial	U	U	U	U	S	U	S	U	S	S	U	U
Shales	U	U	U	U	U	U	U	U	U	U	U	U
Siltstones	U	U	U	U	U	U	U	U	U	U	U	U
Slope deposits	U	U	U	U	U	U	U	U	U	U	U	U

☞

- Click the upper left field in the 2-dimensional table. Press the left arrow (←). The name: `u: Unsuitable` appears. Press the down arrow. You are now in the second field. Press the left arrow. Fill in the entire table. It is faster to work with the arrow key, than to click each field. Select the right name from the list box and click another field. You only have to click the upper field of the next column when you are finished with the last field of the previous column.
- When you are finished, close the table window.

The two-dimensional table is now complete. We can use it in a calculation on the command line. The calculation formula should have the following syntax:

```
Output map=two-dim table[map1,map2]
```

☞

- Type the following formula on the command line of the main window:
  `Gravel = Gravel [ Geology , Landuse ]`
- Press Enter. The Raster Map Definition dialog box is opened.
- Click OK.
- Open the map `Gravel` and check its contents.
- Close the map window.

Summary: Two-dimensional tables

– A two-dimensional table is seen as a matrix, used to define output class names, IDs or values, for each combination of classes/IDs of two input maps.

– Each field in the table (each intersection of a record and a column) represents a specific combination of two classes and or IDs in the two input maps.

– The domain type of the input maps should be either class or ID. The domain type of the two-dimensional table, and of the output map, can be class, id or value.

– Since the output value for each combination of two classes has to be entered manually, this method is mostly used for combining class maps with relatively few classes.

– To apply a two-dimensional table, use the following syntax:

```
output map=two-dim table[map1,map2]
```