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 are part of most spatial analysis processes and 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. In overlay operations new spatial elements are created on the basis of multiple input maps.

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 for combining maps, called Map Calculation. Many maps can be combined at the same time using arithmetic, relational, or conditional operators and many different functions. Map Calculation formulae are 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:\ILWIS 3.0 Data\Users Guide\Chapter08, where the data files for this chapter are stored.

- Double-click the ILWIS icon on the desktop to open ILWIS.
- Use the Navigator to go the directory: C:\ILWIS 3.0 Data\Users Guide\Chapter08.

Introduction to Map Calculation

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

The formulae 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. Data in a dependent map depend on data of 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 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 ILWIS Help topic "Map and Table calculation". Table 8.1 presents an overview of the MapCalc functions.

! You can use the following short-cut keys:
- Ctrl+C (to copy from the Command line to the Clipboard).
- Ctrl+V (to paste information from the Clipboard to the Command line).
This is extremely useful when you have to edit complicated and long formulae.
For an overview of the keyboard shortcuts see the ILWIS Help topic, Appendices Keyboard shortcuts.

! You can also use the history of the Command line. Press the Arrow Up key to retrieve previously used expressions, the Down Arrow key is used to 'scroll' forward again. You can also open the list of previously used commands and expressions by clicking the arrow at the right hand side of the Command line. The amount of commands and expressions that can be retrieved can be set in the Preferences.

In the following exercises, you will see a number of examples of Map Calculation formulae, first some that will produce value maps (section 8.1), then some that will give output maps with a class or ID domain (section 8.3). You will also look at how dependency links can be used to update maps made with Map Calculation (section 8.2).

Table 8.1: Some ILWIS functions used in Map Calculation. All the functions in this table can be used on maps with a domain type value. The Conditional IFF and Undefined functions can also be used on maps with a domain type class or ID. For a complete overview see the ILWIS Help topic "Map and Table calculation".

Functions	Syntax	Operation
Conditional IFF	IFF (*a*,*b*,*c*)	If condition *a* is true, then return the outcome of expression *b*, or else (when condition a is not true) 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 (*a*)	Tests whether *a* is undefined.
	IFUNDEF (*a*,*b*)	If condition *a* is undefined, then return the outcome of expression *b*, else return *a*.
	IFUNDEF (*a*,*b*,*c*)	If condition *a* is undefined, then return the outcome of expression *b*, else return the outcome of expression *c*.
	IFNOTUNDEF (*a*,*b*)	If condition *a* is not undefined, then return the outcome of expression *b*, else return *a*. Tests whether a is undefined.
	IFNOTUNDEF (*a*,*b*,*c*)	If condition *a* is not undefined, then return the outcome of expression *b*, else return the outcome of expression *c*.
Exponential	SQ(*a*)	*a* square; a^2; *a***a*.
	SQ(*a*,*b*)	*a* square plus *b* square; $a^2 + b^2$; (*a***a* + *b***b*).
	SQRT(*a*)	Calculates the positive square root of *a*; $\sqrt{a}$
	HYP(*a*,*b*)	Calculates the positive square root of the sum of *a* square and *b* square; $\sqrt{(a^2 + b^2)}$.
	POW(*a*,*b*)	*a* raised to the power *b*; a^b; The n-th root of *a* is found by: POW(*a*, 1/n).
	EXP(*a*)	Value e (i.e. 2.718) raised to the power *a*; e^a.
Logarithmic	LOG(*a*)	Calculates the 10-based logarithm of *a*; $^{10}\log(a)$
	LN(*a*)	Calculates natural logarithm of *a*; $^{e}\log(a)$
Random	RND(*a*)	Returns random integer values in the range [1;*a*]
	RND(0)	Returns *a* 0 or *a* 1 at random.
	RND( )	Returns random real values in the range [0;1>, i.e. including 0, excluding 1.
Sign	- (*a*)	returns *a* multiplied by –1.
	NEG(*a*)	returns *a* multiplied by –1.
	ABS(*a*)	Returns the absolute (= positive) value of *a*.
	SGN(*a*)	Returns -1 for negative values of *a*, 0 if *a*= 0, and 1 for positive values of *a*.

Table 8.1: (continued)

Functions	Syntax	Operation
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*
NDVI	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.
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 PI2 PIDIV2 PIDIV4 EXP(*a*) %X %Y %L %C	Value π : 3.141592653589793… Value 2π : 6.283185307179586... Value $^1/_2\ \pi$: 1.570796326794896… Value $^1/_4\ \pi$: 0.785398163397448… Returns exponential: e^a ; Value e : 2.718281828459045... 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* , DomainGroup)	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 simplest 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, and not on maps containing classes.

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

Syntax	Operation	Example
+	Add	*a* + *b*
-	Subtract	*a* - *b*
*	Multiply	*a* * *b*
/	Divide	*a* / *b*
^	Exponential operator; POW(*a*,*b*); a^b	*a* ^ b
a MOD *b*	Returns the remainder of *a* divided by *b* (e.g. returns 1 if *a*=10 and *b*=3)	*a* MOD *b*
a DIV *b*	Returns the quotient of *a* divided by *b* (e.g. returns 3 if *a*=10 and *b*=3)	*a* DIV *b*

In the Figure 8.1 some examples of these arithmetic operators are given.

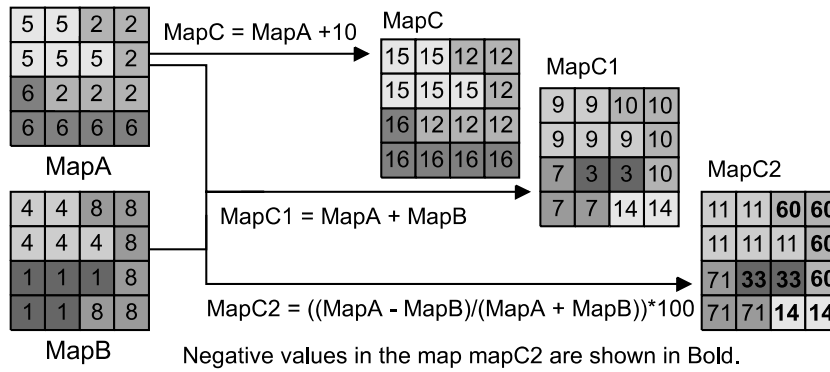

Figure 8.1: Some examples of arithmetic operations in ILWIS. The input maps have domain type value

Let us look at the simplest 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 a constant value of 10.

The second calculation is:
MapC1 = MapA + MapB
This means add the pixel values of MapA and MapB and store the result in MapC1.

The third calculation is:
MapC2 = ((MapA - MapB) / (MapA + MapB)) * 100
This means: Store raster map MapC2, which is the result of the subtraction of MapB from MapA, divided by the sum of MapA and MapB; then multiply this by 100. This formula when applied on two satellite bands (MapB with visible or red values and the MapA with near-infra-red values) is called the NDVI (*Normalized Difference Vegetation Index*). The output values range from -100 to +100.

Relational operators

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

Syntax		Operation	Example
=	eq	Equal to	a = b
<	lt	Less than	a < b
<=	le	Less than or equal to	a <= b
>	gt	Greater than	a > b
>=	ge	Greater than	a >= b
<>	ne	Not equal to	a <> b

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

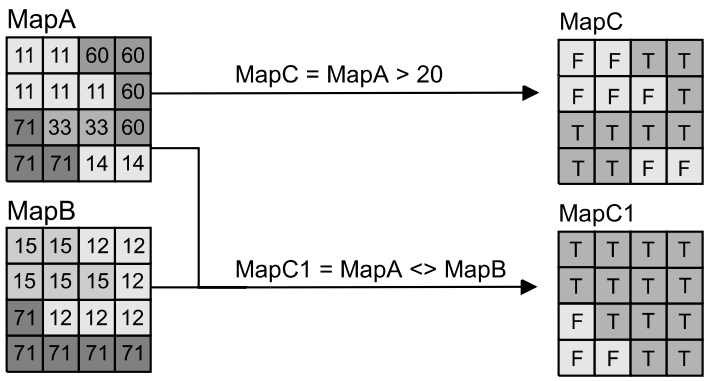

Figure 8.2: Some examples of Boolean statements in ILWIS.

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 (see Figure 8.2):

`MapC = MapA > 20`
This means: For pixels in `MapA` that have a value greater than `20`, the expression is true and `True (1)` is assigned to those pixels in output `MapC`. For pixels where the expression is false, a `False (0)` is assigned (`Bool` domain).

`MapC = MapA <> MapB`
In this statement it is checked whether `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 (1)` or `False (0)`. Such a statement would be useful for change detection, for example to compare two land use maps of different periods.

Logical operators

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

Syntax	Operation	Example
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*)

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`).

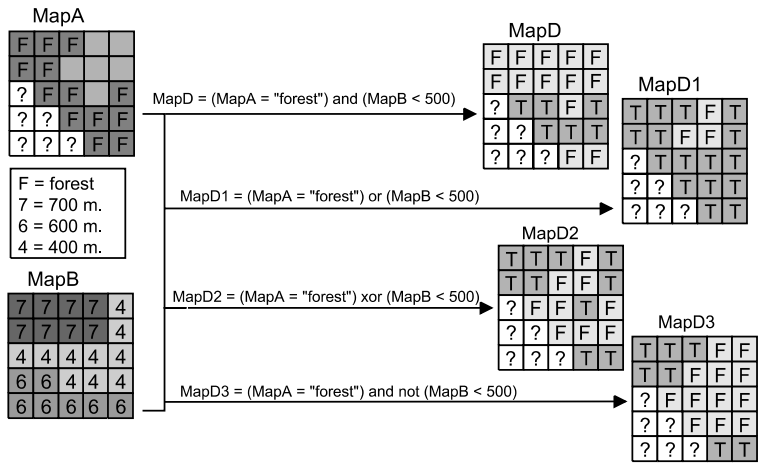

Figure 8.3: Examples of logical operations in ILWIS. MapA has domain type class and MapB has domain type value. The output is either True (1), False (0) or undefined (?).

These operators are also called *Boolean* operators. Examples of Boolean operators (`AND`, `OR`, `XOR`, `NOT`) are presented in Figure 8.3.

`MapD = (MapA="Forest") AND (MapB<500)`

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 True (1) to this pixel in the output map (MapD). Assign value False (0) to all other pixels.

MapD1 = (MapA="Forest") OR (MapB<500)
The expression is true if only 1 of the expressions is true or both of the 2 expressions are true:
- if a pixel in mapA has class name Forest and in MapB not smaller than 500.
- if that pixel in mapB has a value <500 and in MapA not Forest.
- if a pixel in mapA is Forest and if that pixel in MapB <500.

Otherwise the whole expression is false.

MapD2 = (MapA="Forest") XOR (MapB<500)
The expression is true if only 1 of the 2 expressions is true:
- if a pixel in MapA is Forest and in MapB not smaller than 500.
- if a pixel in MapB <500 and in MapA not Forest.

Otherwise the whole expression is false. This statement is called *exclusive OR*.

MapD3 = (MapA="Forest") AND NOT (MapB<500)
When a pixel in MapA has class name Forest and at the same time this pixel in MapB does not have a value less than 500, assign True (1) to this pixel in the output map (return True (1) if the first condition is true and the second is false). Assign False (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:
Output_map = IFF (*Condition, Then Expression, Else Expression*)
or
Output_map := IFF (*Condition, Then Expression, Else Expression*)

Where:

Output_map	Is the name of output map.
=	Is the definition to create a dependent output map.
:=	Is the assignment to create a non-dependent (editable) output map.
IFF	Is the conditional function.
Condition	Is the condition to be met.
Then Expression	Is the calculation that has to be performed when the condition is met.
Else Expression	Is the calculation that has to be performed when the condition is not met.

Some examples of the use of conditional functions in ILWIS are given in Figure 8.4.

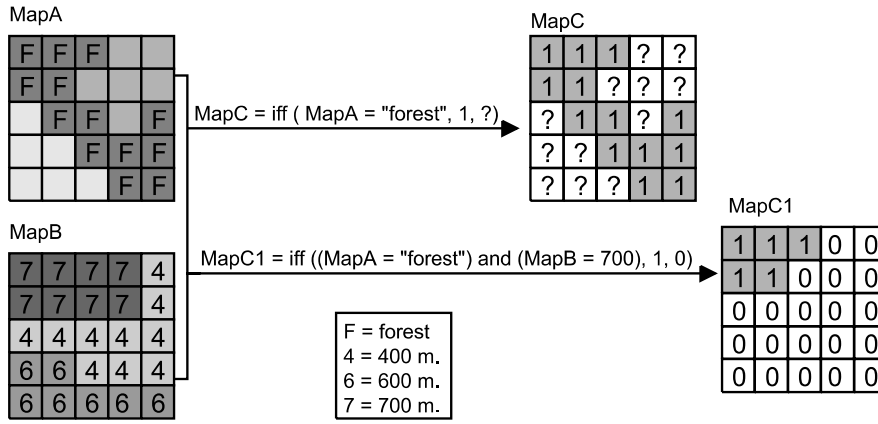

Figure 8.4: Examples of conditional functions in ILWIS. MapA has domain type class and MapB has domain type value.

For simplification purposes, we will not use an expression for the `Then` condition and the `Else` condition but we will simply put a 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 (`?`).

`MapC1 = 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 (`Geomorphology`).

When evaluating the combination of criteria we only look at the criterion, which 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% of the average price. If the same piece of land is also located on a slope more than 20º, which would lead to a decrease of 70% of the average, the value of the land is still 40% of the average, since 40% is less than 70% 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 far too hypothetical - but that you learn to work with Map Calculation formulae. The ILWIS Applications Guide focuses more on applications than on tools.

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

- Select the following maps in the Catalog: `Landuse`, `Slope` and `Geomorphology`; click the right mouse button and select Open Pixel Information.
- Similarly, open the maps `Landuse`, `Slope` and `Geomorphology`.
- To get a quick idea of where the steep slopes are type on the Command line of the Main window:
- `SL = Slope>20` ↵
- To get an idea where the landslides are, type on the Command line of the Main window:
- `LSL =(Geomorphology="OL")OR (Geomorphology="AL")` ↵
- Inspect the values of the maps. Move with the mouse pointer through the `Landuse` map and look at the information of the three maps in the pixel information window.
- Close the map windows 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 GIS analysis with an operation that you are already familiar with: *Reclassification* (see section 7.3). 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 (100*100=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:
- `Landvalue = (Landuse.Landvalue)/25` ↵

- Accept the defaults in the Raster Map Definition dialog box and click Show.
- Display map Landvalue with Representation Pseudo and close it when you have seen the result.

In this formula, you combine a reclassification expression (of the form: map.column) with an arithmetic operator (divide). The attribute table name should not be mentioned unless the table is not linked to the map or to the domain in the Properties sheet.

- Type the following formula:
- `Landvalue1 = IFF (Slope>20, Landvalue*0.7, Landvalue)` ↵

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

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 map `Landvalue1` either have a value which is only 70% of the value as in map `Landvalue`, or the same value as in map `Landvalue`.

- Accept the defaults in the Raster Map Definition dialog box and click Show.
- Display map `Landvalue1` 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% of the average land value. If the pixel is on an active landslide or on an active erosion area, the value is only 40% of the average.

- Type on the Command line of the Main window:
  `Geom = Geomorphology` ↵
- Type the following formula:
  `Landvalue2 = IFF (Geom="OL", Landvalue*0.6, IFF((Geom="AL")OR(Geom="HE"), Landvalue*0.4, Landvalue))` ↵
- Accept the defaults in the Raster Map Definition dialog box and Click Show.

For clarification purposes, we used capitals in the expression but you can type small letters.

! If you get an error message, you may have made a typing error with the brackets or the commas. Use the history (Arrow Up key) to correct your formula. Then press Enter ↵ again.

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. Furthermore, in this formula, codes are used instead of the names of the geomorphologic class names. Unit `"Old landslide"` in the domain `Geomorphology` has code "`OL`". If you use codes, the formulae can be much shorter. Lastly, the relational operator `OR` was used, since both the geomorphologic units `"Active landslide"`, code `"AL"` and `"Heavily eroded area"`, code `"HE"` should give the same decrease of the land values.

Now you have generated two maps each based on one criterion (`Landvalue1`, and `Landvalue2`). What should you do for pixels where more than one of these criteria occur, e.g. pixels with a slope more than 20º, 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 on the Command line of the Main window: `Landval_combined = MIN(Landvalue1, Landvalue2)` ↵
- Click Define in the Raster Map Definition dialog box.
- To visually compare maps `Landvalue` and `Landval_combined`, open both maps with Representation `Inverse` and Stretch `1-15`.
- Use the pixel information window to find out the values in both maps.

Undefined values

Note that the colored areas in map `Landval_combined` occupy a smaller part of the map than in map `Landvalue`. 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 a pixel. In this case a question mark in the map indicates that the part of the area lacks data, and data should be supplied in order to do the analysis properly.
- A pixel is located 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.

- Calculated values in an output map, fall outside the value range defined in the for the output map. For example, when you write the following formula:
`Result = Dem*10`, and select a value range of `0` to `1000` for the output map.
This formula results in a map with undefined values, since the value range of the `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 `IFF` functions. If you use an `IFF` 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 cannot 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 the `AND`, `OR`, `XOR` and `NOT` operators is shown in the *truth tables* below.

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

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

a XOR b	**b=True**	**b=False**	**b=Undefined**
a=True	False	True	Undefined
a=False	True	False	Undefined
a=Undefined	Undefined	Undefined	Undefined

NOT b	**returns:**
b=True	False
b=False	True
b=Undefined	Undefined

So for example, if we have the following IFF 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 IFF 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.

- If you did not notice before that the lower right part of the map Landval_combined seems to be missing, display map Geom next to the other maps and check.
- Type the following formula on the Command line of the Main window: Landval_final = IFUNDEF(Geom, Landvalue1, Landval_combined) ↵

The IFUNDEF function tests whether the condition part, i.e. map Geom, is undefined. In words this formula means: If a pixel in map Geom is undefined (which means no information on geomorphology is available), than we take the value from the map Landvalue1 (which includes the slope related land prices), otherwise the value from the map Landval_combined.

In ILWIS the function IFF(ISUNDEF(*a*), *b*, *c*) gives the same result as the expression IFUNDEF(*a*, *b*, *c*).

- Accept the defaults in the Raster Map Definition dialog box and click Show.
- Display the map Landval_final with Representation Inverse, Stretch 1-15. 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.

- MapCalc formulas are typed on the Command line of the Main window.

- MapCalc formulas use raster maps as input and produce a raster map as output.

- There is a wide range of operators and functions that can be used:
  Arithmetic operators to multiply, divide, raise to a power, 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 than, smaller than, equal to another expression etc.

- Most functions are listed in Table 8.1.

- The most important function is the `IFF` function, which has the following structure: `IFF` (*Condition, Then Expression, Else Expression*).

- `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.

- Pixels in an output map are undefined if the pixel was already undefined in any of the input maps. You don't know the result of a formula if one of the operands is undefined; this depends also on the operators that are used.

- The functions `IFF(ISUNDEF(`*a*, *b*, *c*`)` and `IFUNDEF(`*a*, *b*, *c*`)` test whether expression *a* is undefined. You can use these functions 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 `Geomorphology`, `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) five *dependent maps* were made: `Landvalue`, `Landvalue1`, `Landvalue2`, `Landval_combined`, and `Landval_final`.

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 or not. As soon as one of any of the input maps changed, the dependent map knows that it is no longer up-to-date.

- Open the Properties of map `Landval_final`.
- On the Dependency tab one can read: Object is up-to-date. This means that no changes were made in the input maps after the time that the map `Landval_final` was created.
- Close the Properties of the map `Landval_final` by clicking the Cancel button.
- Move with the mouse pointer over map `Landval_final`. On the Status bar you see on the right-hand side D, C, U. This means that map `Landval_final` is Dependent (D), Calculated (C) and Up-to-Date (U).

! Another way to see if an object is Dependent (D), Calculated (C) and/or Up-to-Date (U) is by switching the Catalog to Details View (by choosing the Details command from the View menu or by clicking the Details button in the Standard toolbar of the Main window).

- Open 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`.

- Display map Landuse.
- In the attribute table Landuse, edit the Landvalue field for the Lake record. Type 400.
- Double-click the Lake in the map window; the landvalue of the lake will now be 400. Close the map and table when done.
- Open map Landval_final and open a pixel information window with maps Landuse and Landval_final.
- Check the value of the lakes.

As you can see the lakes in the map Landval_final 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 Landval_final and the pixel information window.
- Open the Properties of raster map Landval_final.
- On the Dependency tab of the raster map Landval_final one can now read:

  Object is not up-to-date: Column Landuse.Landvalue (day, month, year, time).
  It is indicated (day, month, year, time) when you have updated the column Landvalue (if it is correct that should be about a minute ago).
- Go to the Catalog and move with the mouse pointer over map Landval_final.
- On the Status bar you see the characters D, C, N. This means that the map Landval_final is Dependent (D), Calculated (C) but that the map is Not Up-to-Date (N).
- On the Depencency tab, press the Make Up-to-Date button. The Check Up-to-date dialog box appears.
- Click Yes to answer the question: Map Calculate "Landval_final" 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 Landval_final. In fact it will do the previous exercise for you again. First the map Landvalue is recalculated, in which the column Landvalue from the table Landuse was used (the one that you just updated). Then the maps Landvalue1, and Landvalue2 are recalculated. These are combined in the map Landval_combined, after which the undefined values are removed and the final map Landval_final is made. The entire recalculation may take a minute. You will see the progress bars of the various calculations. When the progress bars disappear, the calculation is finished.

- Move with the mouse pointer over map `Landval_final`. On the Status bar you see again the characters D, C and U.
- Open raster map `Landval_final` and click the location of one of the lakes. As you can see the map `Landval_final` is now updated.
- Close raster map `Landval_final`.

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 depend on data of input maps via the used formula.

- When data in a source map is changed, this is indicated in the Properties sheet of the dependent map. The map is then no longer up-to-date.

- If a final output map is no longer up-to-date, all dependent maps in between an edited source map and a final output map, will be recalculated when you press the Make Up-to-Date button. For the final output map a chain of recalculations will take place until the final output map is up-to-date again.

- The dependency of maps allows for easy updating.

8.3 Map Calculation formulas resulting in class or ID maps

When we work with IFF 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 IFF function one of the last three possibilities, the program will suggest that you use an existing domain, unless two new names are in the IFF function. When you decide to use the default existing domain and press OK in the Column Properties and/or in the Raster Map Definition 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 (AL = Active Landslide, OL = Old LAndslide) you can type the following formula:

- Type the following formula on the Command line of the Main window:
  `Slide = IFF((Geom="AL")OR(Geom="OL"), Geom, "?")` ↵

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

- Accept the defaults by clicking the Show button in the Raster Map Definition dialog box.
- Check the contents of map Slide and close it afterwards.

In an expression, you can use different maps with different domains. Suppose you want to find out the geological unit of the landslides that are between 3500 and 4000 meters. You can use the history of the Command line to adapt the previously used formula.

- Type the following formula on the Command line of the Main window: `Slide1 = IFF (((Geom="AL")OR(Geom="OL"))AND INRANGE(Dem, 3500, 4000), Geology, "?")` ↵
- Click Show in the Raster Map Definition dialog box.

In the condition 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 meters.

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 `IFF` 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 of the Main window: `Slide2 = IFF((Geom="AL")OR(Geom="OL"), Geom, "No landslide")` ↵
- The Raster Map Definition dialog box indicates the default domain `Geomorphology`.
- Click Show.

Now the Merging Domains dialog box appears with the question: Add string "No landslide" to domain "Geomorphology".

If you answer No, the name "No landslide" will not be added as a class to the domain Geomorphology. Since the name "No landslide" is not in the Geomorphology 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 landslide" is added to domain Geomorphology.

- Answer Yes to the question: Add string "No landslide" to domain "Geomorphology".
- Inspect the results in map Slide2.
- Then 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 of the Main window: Slide3 = IFF ((Geom="AL")OR(Geom="OL"), "Landslide", Geom) ↵
- The Raster Map Definition dialog box indicates the default domain Geomorphology.
- Click Show.
- Answer the question: Add string "Landslide" to domain "Geomorphology" with Yes.
- Inspect the results in map Slide3. You can change the color of unit Landslide by double-clicking the Landslide item in the legend (Layer Management pane).
- Close the map window when done.

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 (OL), Active landslide (AL) and Heavily eroded area (HE) to the map Landuse.

- Type the following formula on the Command line of the Main window: `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 Class 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 Class editor. Now you are back in the Raster Map Definition dialog box. You will see that the domain is now `Slide4`.
- Click Show.
- Answer the question: `Merge strings of domain "Geomorphology" into domain "Slide4"` with Yes.
- Answer the question: `Merge strings of domain "Landuse" into domain "Slide4"` with Yes.
- Inspect the results in map `Slide4` and in domain `Slide4`.
- Close the map and the domain afterwards.

Now the two domains `Geomorphology` 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"`.

- Type the following formula on the Command line of the Main window: `Slide5 = IFF((Geom="AL")OR(Geom="OL"), "Landslides", "No landslides")` ↵
- In the Raster Map Definition dialog click the Create Domain button. The Create Domain dialog box appears.
- Type `Slide5` for the Domain Name. Make sure the option Class is selected and click OK. The Domain Class editor is opened.
- Press the Add Item button on the toolbar. The Add Domain Item dialog box is opened. ➥

- Add the class "Landslides" and click OK.
- Press the Insert-key and add the class "No landslides".
- Close the Domain Class editor. You are back in the Raster Map Definition dialog box.
- Type the Description: Presence of Landslides and click Show.
- Inspect the results in map Slide5 and domain Slide5 and close the map window and editor afterwards.

You could have created a new domain beforehand as well. 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". In that case the formula would be:
Slide6 = (Geom="AL")OR(Geom="OL") ↵
Select 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 formulae 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 are 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-table* and an *output cross-map*. The output cross-table and the output cross-map obtain an ID domain with the same name as the output cross-table. The domain contain 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 8.5.

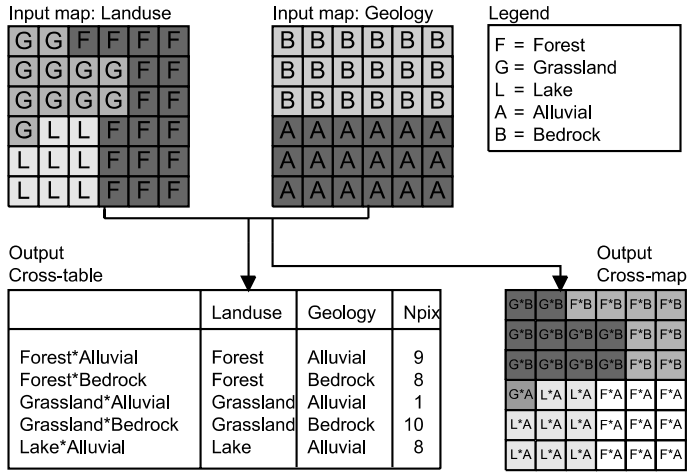

Figure 8.5: Example of a cross operation. The maps Landuse and Geology are combined. A cross table and a cross map are made.

In the following exercise you will do two cross operations:

- Between two class maps: `Landuse` and a classified slope map `Slope_classes`, in order to find the percentages of flat, moderate and steep slopes for each land use type.
- Between an identifier map `Catchment` 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 `Slope_classes` with three classes: `Flat` (0-10º), `Moderately steep` (10-25º), and `Steep` (> 25º). Note that some examples of classifying value maps were already presented in section 7.5.

- Display the maps `Landuse` and `Slope_classes` and check the meaning of the units. Close the map windows.
- Open the Raster Operations item in the Operation-tree and double-click the Cross operation. The Cross dialog box is opened.
- Select raster map `Landuse` in the list box 1st Map.
- Select raster map `Slope_classes` in the list box 2nd Map.
- Type `Landuse_slope` in the text box Output Table.
- Type for the Description: `Combination of Landuse and Slope_classes`.
- Select the Output Map check box.
- Type `Landuse_slope` in the text box Output Map.
- Click Show. The output cross table is displayed.

It contains quantitative information on the combinations of the two input maps. You can increase the display width of the domain of the table by dragging the first column header separator to the right.

- Display the output cross map `Landuse_slope` 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 `Slope_classes`.
- Close the map `Landuse_slope`.
- 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 of 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 `Totalarea` of table `Landuse`.

- Type the following formula on the Command line of the table window: `Flat = IFF(Slope_classes="Flat", Area, 0)` ↵
- The Column Properties dialog box is opened. Click OK.
- Press the Arrow Up-key and edit the previous formula, so that it is: `Moderate = IFF(Slope_classes="Moderate", Area, 0)` ↵
- The Column Properties dialog box is opened. Click OK.
- Press the Arrow Up-key and edit the previous formula, so that it is: `Steep = IFF(Slope_classes="Steep", Area, 0)` ↵
- The Column Properties dialog box is opened. Click OK.
- 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 Output Table check box and type the table name: `Landuse`. Type for the Output Column: `Totalarea`.
- Click OK in the Aggregate Column dialog box. Column `Totalarea`, containing area sums per landuse class will be added to table `Landuse`.
- 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 Output Table check box, 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 `Totalarea`, `Flat`, `Moderate` and `Steep`.
- Close the table `Landuse_slope`.
- Open the table `Landuse`. As you can see it contains the columns `Totalarea`, `Flat`, `Moderate` and `Steep` that you just generated.

Now you will calculate the percentage landuse found on flat, moderate and steep slopes.

- Type the following formula on the Command line of the table window: `Pflat = 100*(Flat/Totalarea)` ↵
- The Column Properties dialog box is opened.
- Select the Domain `Perc` (percent) and enter for the Precision `1.0`. Click OK.
- Create also the columns `Pmoderate` and `Psteep` with similar formulas. Select Domain `Perc` for these columns and use a Precision of `1.0`.
- Evaluate whether the results make sense and close the table `Landuse` when done.

The columns Pflat, Pmoderate 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 analyses, 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 Catchment (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 (section 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 Catchment in the list box 1st Map.
- Select raster map Drainage in the list box 2nd Map.
- Type Catchment_drainage in the text box Output Table.
- Type for the Description: Cross of catchment and drainage length per pixel. Click Show.

You can inspect the Properties of the table to find out the number of records in the cross-table.

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 `Catchment_drainage`:
  `Length = Drainage * npix` ↵
- The Column Properties dialog box appears. Accept the defaults and click OK.
- 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 `Catchment`, select the Output Table check box, and type the table name: `Catchment`. Type for the Output Column: `Totallength`.
- 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 `Catchment`. You can open table `Catchment` to check this; then close the table. Now you also need to know the total area of each catchment. The output column `Totalarea` is calculated by using the `Sum` aggregation function on the column `Area` grouped by the catchments (column `Catchment`).

- 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 `Catchment`, select the Output Table check box, and type the table name: `Catchment`. Type for the Output Column: `Totalarea`.
- Click OK in the Aggregate Column dialog box.
- Close table `Catchment_drainage`.

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 table `Catchment`.
- Type the following formula on the Command line of the table window:
  `Drainagedensity = 1000*Totallength/Totalarea` ↵
- The Column Properties dialog box is opened. Click OK.

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 table `Catchment`.

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/IDs 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-table and a cross-map. Both output objects obtain an ID domain with the same name as the output cross-table. The ID 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.

- The cross-map is often less important because it contains too many different combinations. The crossing operation is mostly done to obtain cover information of one map with respect to another, and not so much to generate a new combination map.

8.5 Two-dimensional tables

As could be seen in the previous exercise, crossing of 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/IDs in the two input maps (see Figure 8.6). 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 8.6), the fields in the two-dimensional table use a class domain with two classes `Suitable`, and `Unsuitable`. The two class domains `Geology` and `Landuse` define the records and columns in the table.

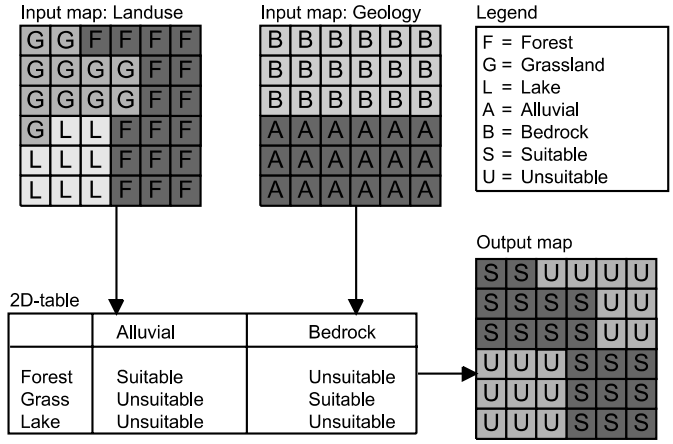

Figure 8.6: Example of the use of a two-dimensional table, used to combine two input maps. The user can determine for each combination of classes in the input maps, what will be the result in the output map. In this example Forest and Alluvial, and Grass and Bedrock result in Suitable.

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 map `Cityblock`, with 717

identifiers. If we would use it in combination with the Landuse 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 for 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 button next to the drop-down list box of the domain. The Create Domain dialog box is opened.
- Type for the Domain Name: Gravel. Select Type Class. Type the Description: Suitability for gravel extraction and click OK. The Domain Class editor is opened.
- Press the Insert-key, and enter for the Name: Suitable, and the Code: S. Click OK.
- Press the Insert-key, and enter for the Name: Unsuitable, and the Code: U. Click OK.
- Close the Domain Class 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 units (rows) and land use units (columns) (U = unsuitable, S= suitable).

	Ag	Ai	Ap	Br	Bs	Fo	Gr	La	Ri	Sh	Uc	Up
Gl	U	U	U	U	S	U	S	U	S	S	U	U
Ld	U	U	U	U	U	U	U	U	U	U	U	U
Oa	U	U	U	U	S	U	S	U	S	S	U	U
Qu	U	U	U	U	U	U	U	U	U	U	U	U
Ra	U	U	U	U	S	U	S	U	S	S	U	U
Sh	U	U	U	U	U	U	U	U	U	U	U	U
Si	U	U	U	U	U	U	U	U	U	U	U	U
Sd	U	U	U	U	U	U	U	U	U	U	U	U

- Select a rectangle for all the fields in the first 4 columns.
- Click the right mouse button in the selection, and choose Edit from the context-sensitive menu. The Edit selected field(s) dialog box appears.
- In the Edit selected field(s) dialog box select the class `Unsuitable`.
- Subsequently, edit all the fields in the columns `Forest`, `Lake`, `Urban centre` and `Urban periphery` to `Unsuitable`.
- Edit the fields of column `Bare soils` one by one. Then select all fields of column `Bare soils` and use the Copy and Paste (on the context-sensitive menu) to paste the values into columns `Grassland`, `Riverbed` and `Shrubs`.
- 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 of the Main window. The calculation formula should have the following syntax:

`Output_map` = *two-dim_tablename* [*map1*, *map2*]

Where:

two-dim_tablename	is the name of the two-dimensional table
map1	corresponds with the records in the 2D table
map2	corresponds with the columns in the 2D table

Make sure that you use square brackets [ ]instead of curly brackets ( )

- Type the following formula on the Command line of the Main window:
  `Gravel = Gravel[Geology, Landuse]` ↵
- The Raster Map Definition dialog box is opened.
- Click Show.
- Check the contents of map `Gravel`.
- Close the map window when done.

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_tablename [map1, map2]`