| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [ArbitraryModeFlow/src](http://docs.google.com/index.html) - ArbitraryModeCellContainer.cpp (source / [functions](http://docs.google.com/ArbitraryModeCellContainer.cpp.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 54 | 82 | 65.9 % | | Date: | 2012-03-23 |  | Functions: | 11 | 11 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 13 | 44 | 29.5 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : :   2 : : #include <boost/ref.hpp>  3 : :   4 : : #include "ArbitraryModeCellContainer.h"  5 : : #include "ArbitraryModeShotDistributor.h"  6 : : #include "ConverterParams.h"  7 : : #include "dec\_types.h"  8 : : #include "pattern.h"  9 : : #include "PatternProcessor.h"  10 : : #include "SDDB.h"  11 : : #include "print.h"  12 : : #include "HealthCheckCounter.h"  13 : :   14 : : namespace nft {  15 : :   16 : 2430 : PatternExploder ArbitraryModeCellContainer::mExploder;  17 : :   18 : : void  19 : 63 : ArbitraryModeCellContainer::setSFT(SFTable::Pointer value) {  20 : 63 : mSFT = value;  21 : : }  22 : :   23 : : void  24 : : ArbitraryModeCellContainer::process(  25 : : PatternProcessor & inPatternProcessor,  26 : 63 : ArbitraryModeShotDistributor & inShotDistributor) throw(DPL::Error \*) {  27 : :   28 : 63 : HealthCheckCounter::instance()->incrCounter(  29 : : HealthCheckCounter::eUpdateFrequent |  30 : : HealthCheckCounter::eUpdateInfrequent |  31 : : HealthCheckCounter::eUpdateSeldom );  32 : :   33 : 63 : ConverterParams \*pParams = ConverterParams::instance();  34 : 63 : inShotDistributor.setTargetCellContainer( this );  35 : 63 : inPatternProcessor.registerDistributor( boost::ref(inShotDistributor) );  36 : 63 : int nX = 0, nY = 0, eX = 0, eY = 0;  37 : 63 : GroupContext tmpPGC = \*(mTargetCell->pgc);  38 : 63 : std::vector<PatternPosition> positions;  39 : 63 : SDDB \*pSDDB = SDDB::instance();  40 : 149 : while(true)  41 : : {  42 : 149 : EP newEP;  43 : 149 : try  44 : : {  45 [ + + ]: 149 : if( -1 == vsbGetEP( &tmpPGC, &newEP, &nX, &nY, &eX, &eY ) )  46 : : {  47 : 63 : break;  48 : : }  49 : : }  50 [ # # ]: 0 : catch(DPL::Error \*pChild)  51 : : {  52 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  53 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  54 : : ErrorHandler::eErrFunction,  55 : : pErr->getErrorMsg( ErrorHandler::eErrFunction,  56 : : "vsbGetEP"));  57 : 0 : return;  58 : : }  59 : 86 : positions.clear();  60 : 86 : ParsedPattern pat;  61 : : //(\*(cell->pgc)).ppc = tmpPGC.ppc;  62 : 86 : try  63 : : {  64 : 86 : mExploder.getParsedPattern(  65 : : &tmpPGC, positions, newEP, nX, nY, eX, eY,  66 : : mTargetCell->startPos, pat );  67 : : }  68 [ # # ]: 0 : catch( DPL::Error \*pChild )  69 : : {  70 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  71 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  72 : : ErrorHandler::eErrFunction,  73 : : pErr->getErrorMsg(  74 : : ErrorHandler::eErrFunction,  75 : : "PatternExploder::getParsedPattern"));  76 : 0 : return;  77 : : }  78 : 86 : try  79 : : {  80 : 86 : pSDDB->EPGenerationCompleted(positions.size());  81 : : }  82 [ # # ]: 0 : catch( DPL::Error \*pChild )  83 : : {  84 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  85 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  86 : : ErrorHandler::eErrFunction,  87 : : pErr->getErrorMsg(ErrorHandler::eErrFunction,  88 : : "SDDB::EPGenerationCompleted"));  89 : 0 : return;  90 : : }  91 : 86 : try  92 : : {  93 : 86 : pSDDB->ParsedPatternGenerated(mTargetCell, pat, positions);  94 : : }  95 [ # # ]: 0 : catch( DPL::Error \*pChild )  96 : : {  97 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  98 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  99 : : ErrorHandler::eErrFunction,  100 : : pErr->getErrorMsg(  101 : : ErrorHandler::eErrFunction,  102 : : "SDDB::ParsedPatternGenerated"));  103 : 0 : return;  104 : : }  105 : :   106 : :   107 : : uint32\_t numLayers  108 : 86 : = pParams->getStartupMessage()->uiSFLCountInDpb;  109 : 86 : for(std::vector<PatternPosition>::iterator it = positions.begin();  110 : : it != positions.end(); ++it )  111 : : {  112 : 149 : pat.setPosition( \*it );  113 [ + - ][ + + ]: 318 : for( uint32\_t i = 0; i < numLayers; i++ )  114 : : {  115 : 169 : pSDDB->setLayerNumber(i);  116 : 169 : try  117 : : {  118 : 169 : pSDDB->ParsedPatternReadyForProcessing(pat);  119 : : }  120 [ # # ]: 0 : catch( DPL::Error \*pChild )  121 : : {  122 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  123 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  124 : : ErrorHandler::eErrFunction,  125 : : pErr->getErrorMsg(  126 : : ErrorHandler::eErrFunction,  127 : : "SDDB::ParsedPatternReadyForProcessing"));  128 : 0 : return;  129 : : }  130 : :   131 : 169 : try  132 : : {  133 : 169 : inPatternProcessor.process( pat,i );  134 : : }  135 [ # # ]: 0 : catch( DPL::Error \*pChild )  136 : : {  137 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  138 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  139 : : ErrorHandler::eErrFunction,  140 : : pErr->getErrorMsg(  141 : : ErrorHandler::eErrFunction,  142 : : "PatternProcessor::process"));  143 : 0 : return;  144 : : }  145 : 169 : try  146 : : {  147 : 169 : pSDDB->ParsedPatternProcessingCompleted();  148 : : }  149 [ # # ]: 0 : catch( DPL::Error \*pChild )  150 : : {  151 : 0 : ErrorHandler \*pErr = ErrorHandler::instance();  152 : 0 : pErr->errorAdd(pChild, \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_,  153 : : ErrorHandler::eErrFunction,  154 : : pErr->getErrorMsg(  155 : : ErrorHandler::eErrFunction,  156 : : "SDDB::ParsedPatternProcessingCompleted"));  157 : 0 : return;  158 : : }  159 : : }  160 : : }  161 : : }  162 : : }  163 : :   164 : : void  165 : 63 : ArbitraryModeCellContainer::setIsComplete(bool value) {  166 : 63 : mIsComplete = value;  167 : : }  168 : : void  169 : : ArbitraryModeCellContainer::setCellPosition( const IntPosition &inPosition )  170 : 63 : {  171 : 63 : mCellPosition = inPosition;  172 : : }  173 : :   174 : : void  175 : 63 : ArbitraryModeCellContainer::setTargetCell( Cell \*inCell ) {  176 : 63 : mTargetCell = inCell;  177 : : }  178 : :   179 : : ArbitraryModeCellContainer::ArbitraryModeCellContainer()  180 : 63 : :mSFT(),  181 : : mTargetCell(),  182 : : mIsComplete(),  183 : 63 : mCellPosition() {  184 : : }  185 : :   186 : : ArbitraryModeCellContainer::~ArbitraryModeCellContainer()  187 : 126 : {  188 : 63 : SFLayer::Pointer layers = mSFT->getSFLayers();  189 : : uint32\_t numLayers = ConverterParams::instance()  190 : 63 : ->getStartupMessage()->uiSFLCountInDpb;  191 [ # # ]: 146 : for(SFLayer::Pointer layer = layers; layer != layers + numLayers; layer++ )  [ # # + - ]  [ + + ]  192 : : {  193 : 83 : layer->releaseSFGroups();  194 : : }  195 [ # # ][ # # ]: 146 : delete []layers;  [ + - ][ + + ]  196 [ # # ][ + - ]: 63 : delete mSFT;  197 [ # # ][ + - ]: 63 : delete mTargetCell;  198 : : }  199 : :   200 : 1215 : } // namespace nft |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |